Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

On simplification of schema mappings

Diego Calvanese [a,*], Giuseppe De Giacomo [b], Maurizio Lenzerini [b], Moshe Y. Vardi [c]

[a] *KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Piazza Domenicani 3, I-39100 Bolzano, Italy*
[b] *Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma, Via Ariosto 25, I-00185 Roma, Italy*
[c] *Dep. of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77251-1892, USA*

ARTICLE INFO

Article history:
Received 3 October 2011
Received in revised form 3 July 2012
Accepted 16 January 2013
Available online 21 January 2013

Keywords:
Schema mappings
Mapping simplification
LAV
GAV
GLAV
Graph databases
Regular path queries

ABSTRACT

A schema mapping is a formal specification of the relationship holding between the databases conforming to two given schemas, called source and target, respectively. While in the general case a schema mapping is specified in terms of assertions relating two queries in some given language, various simplified forms of mappings, in particular LAV and GAV, have been considered, based on desirable properties that these forms enjoy. Recent works propose methods for transforming schema mappings to logically equivalent ones of a simplified form. In many cases, this transformation is impossible, and one might be interested in finding simplifications based on a weaker notion, namely logical implication, rather than equivalence. More precisely, given a schema mapping M, find a simplified (LAV, or GAV) schema mapping M' such that M' logically implies M. In this paper we formally introduce this problem, and study it in a variety of cases, providing techniques and complexity bounds. The various cases we consider depend on three parameters: the simplified form to achieve (LAV, or GAV), the type of schema mapping considered (sound, or exact), and the query language used in the schema mapping specification (conjunctive queries and variants over relational databases, or regular path queries and variants over graph databases). Notably, this is the first work on comparing schema mappings for graph databases.

1. Introduction

A schema mapping is a formal specification of the relationship holding between the databases conforming to two given schemas. Many papers in the last decade point out the importance of schema mappings in several data management tasks, especially those requiring inter-operability between different information systems, such as data integration [1,2], data exchange [3,4], and model management [5].

In data integration, schema mappings are established between the source schema and the global schema, also called mediated schema. In this context, schema mappings are used by the query processor when planning the accesses to the source data for answering queries posed in terms of the global schema. In data exchange, schema mappings are specified in terms of a source schema and a target schema, and determine how the source data should be transferred to the target in order to populate a database conforming to the target schema. Schema mappings are also the main objects of interest

[*] Corresponding author. Fax: +39 0471016009.
E-mail addresses: calvanese@inf.unibz.it (D. Calvanese), degiacomo@dis.uniroma1.it (G. De Giacomo), lenzerini@dis.uniroma1.it (M. Lenzerini), vardi@cs.rice.edu (M.Y. Vardi).
URLs: http://www.inf.unibz.it/~calvanese/ (D. Calvanese), http://www.dis.uniroma1.it/~degiacomo/ (G. De Giacomo), http://www.dis.uniroma1.it/~lenzerini/ (M. Lenzerini), http://www.cs.rice.edu/~vardi/ (M.Y. Vardi).

0022-0000/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcss.2013.01.005

in model management, whose goal is to support the creation, compilation, reuse, evolution, and execution of mappings between schemas, expressed in a wide range of models.

A schema mapping is constituted by two schemas and a set of mapping assertions between the two. We follow the data exchange terminology, and call the two schemas *source* and *target*, respectively. As usual, we assume that each assertion relates a query q_s expressed over the source schema to a query q_t expressed over the target schema, and specifies a correspondence between the tuples computed by q_s on the source databases and those computed by q_t on the target database. In the following, we consider two types of schema mappings, called *sound* and *exact*, respectively. The correspondence specified by assertions in a sound mapping is *inclusion*, whereas the correspondence specified by assertions in an exact mapping is *equality*. Semantically, a schema mapping M specified on source schema S and target schema T is characterized by the set of pairs $(\mathcal{D}_s, \mathcal{D}_t)$ such that $\mathcal{D}_s$ is a source database, i.e., a database conforming to S, $\mathcal{D}_t$ is a target database, i.e., a database conforming to T, and $\mathcal{D}_s$ and $\mathcal{D}_t$ satisfy the correspondences sanctioned by the assertions in M, denoted as $(\mathcal{D}_s, \mathcal{D}_t) \models M$.

Although mappings of general form are specified using arbitrary queries, various restricted forms of mappings have been considered in this investigation. Notable cases of restricted forms of mappings are "Local-As-Views" (LAV) and "Global-As-Views" (GAV) mappings [1]. In LAV mappings, the source queries in the assertions are constituted by one atom, and exactly one assertion appears for each relation symbol in the source schema. In other words, a LAV mapping associates to each element of the source schema one view over the target schema. Conversely, a GAV mapping associates to each element of the target schema one view over the source schema. Extending the above terminology, the term GLAV is often used to refer to unrestricted forms of schema mappings.

Since the pioneering work in data integration [6], schema mappings have been widely investigated in the last years. Recently, several research works have been focused on principles and tools for comparing both schema mapping languages, and schema mappings expressed in a certain language [7–12]. In [10], schema mapping optimization is studied, based on logical equivalence. A set of optimality criteria are proposed for an important class of relational schema mappings, and rewriting rules for transforming a schema mapping into an equivalent optimal one are presented. Notably, LAV and GAV enjoy many of the optimality criteria mentioned in the paper. It follows that the proposed rewriting rules often lead to transforming an input schema mapping into one of the two simplified forms.

Most of the work on optimization and simplification of schema mappings has concentrated so far on transformations preserving logical equivalence. However, there are cases where equivalence preserving simplification is not possible, as demonstrated for LAV by Example 1.1 below. To address such cases, we argue that simplification should be based on a weaker notion, namely logical implication, rather than equivalence. A mapping M *logically implies* a mapping M', denoted $M \models M'$, if whenever $(\mathcal{D}_s, \mathcal{D}_t) \models M$, we also have that $(\mathcal{D}_s, \mathcal{D}_t) \models M'$.

Example 1.1. We restrict our attention to schema mappings constituted by mapping assertions using positive queries. Consider a source schema S consisting of the relations $r_1/3$ and $r_2/2$ and a target schema T consisting of the relations $t_1/3$ and $t_2/2$. Notice that a LAV mapping from S to T has the form

$$\begin{aligned}&\bigl\{(x,y,z)\bigm| r_1(x,y,z)\bigr\} \rightsquigarrow q_1,\\ &\bigl\{(x,y)\bigm| r_2(x,y)\bigr\} \rightsquigarrow q_2\end{aligned}$$

where q_1 and q_2 are two queries over T with arity three and two respectively. Consider now a schema mapping M consisting of the following mapping assertion

$$\bigl\{(x,w,z)\bigm| r_1(x,w,y)\wedge r_2(y,z)\bigr\} \rightsquigarrow \bigl\{(x,w,z)\bigm| t_1(x,w,v)\wedge t_1(v,w,e)\wedge t_2(e,z)\bigr\}.$$

Suppose that M is interpreted under the sound semantics. Now, let $\mathcal{D}_s$ be a database such that $r_1^{\mathcal{D}_s} \neq \emptyset$, and $r_2^{\mathcal{D}_s} = \emptyset$, and let $\mathcal{D}_t$ be the empty database. Clearly, $(\mathcal{D}_s, \mathcal{D}_t) \models M$. On the other hand, for any sound LAV mapping M' on the same alphabet as M, it holds that $(\mathcal{D}_s, \mathcal{D}_t) \not\models M'$, because $r_1^{\mathcal{D}_s} \neq \emptyset$, while the query $M'(r_1)$ that M' associates to r_1 is such that $M'(r_1)^{\mathcal{D}_t} = \emptyset$, and therefore the assertion $r_1 \rightsquigarrow M'(r_1)$ in M' cannot be satisfied by $(\mathcal{D}_s, \mathcal{D}_t)$. It follows that no LAV mapping M' exists such that $M \models M'$, and therefore equivalence preserving LAV simplification of M is impossible to achieve.

Example 1.2. Refer again to the schema mapping M of Example 1.1, and consider the sound LAV mapping M'' constituted by the following two mapping assertions:

$$\begin{aligned}&\bigl\{(x,w,y)\bigm| r_1(x,w,y)\bigr\} \rightsquigarrow \bigl\{(x,w,y)\bigm| t_1(x,w,v)\wedge t_1(v,w,y)\bigr\},\\ &\bigl\{(x,y)\bigm| r_2(x,y)\bigr\} \rightsquigarrow \bigl\{(x,y)\bigm| t_2(x,y)\bigr\}.\end{aligned}$$

It is not difficult to see that $M'' \models M$. Therefore, if we are happy with LAV simplification based on logical implication rather than logical equivalence, M'' represents an acceptable simplification of M.

Mapping simplification based on logical implication is the subject of this paper. The basic problem we consider can be stated as follows: given a schema mapping M, check whether a simplified (LAV, or GAV) schema mapping M' exists such that M' logically implies M (and, if it exists, find one). We formally introduce this problem, and study it in a variety of cases, depending on three parameters:

1. the simplified form to achieve, namely, LAV or GAV,
2. the type of schema mapping considered, namely, sound or exact, and
3. the data model and the query language used in the schema mapping specification.

As for the first parameter, we essentially concentrate on LAV in this paper. We discuss GAV only briefly, pointing out that GAV simplification is an open problem in several cases.

As for the type of mapping, although the sound semantics is the most popular one in data exchange [3], the importance of considering exact schema mappings is widely recognized for both data exchange [13], and data integration [14].

As for the data model and the query language used in schema mappings, we consider both the relational data model with conjunctive queries and unions thereof, and the graph database model with regular path queries and their extensions. Note that, while schema mappings have been extensively studied for relational data, and, to some extent, for XML data [15], this is the first paper on comparing schema mappings for graph databases. Graph databases [16] were introduced in the 80's, and are regaining wide attention recently [17–19], for their relevance in areas such as semi-structured data, biological data management, social networks, and the semantic web.

The notion of simplification put forward in our work can be seen as a form of approximation: Given a schema mapping M (of a general form), find a LAV schema mapping M' "approximating" M. It is well known that there are at least two ways of approximating a logical theory T: one way is through a theory T_1 that is *stronger* than T, i.e., such that T_1 logically implies T, and the other way is through a theory T_2 that is *weaker* than T, i.e., such that T logically implies T_2. For example, in the work on Horn approximation of propositional theories, both kinds of approximations have been considered (see, [20,21]). In this paper, we study one of the two approximations, namely the one where we look for a simplified LAV/GAV schema mapping that is stronger (or, more precisely, not weaker) than the given schema mapping. Note that all inferences obtained by means of the original mapping are also obtained by means of the stronger mapping. At the same time, with this form of simplified mapping we can infer more than with the original mapping. On the other hand, with a weak approximation, we would get the opposite: all inferences obtained by means of the simplified mapping are also valid inferences with respect to the original mapping. The weak forms of approximation are also of interest, but are not investigated here and are left for future study.

The results we present in this paper can be summarized as follows. We first illustrate our ideas with relational mappings, where the results follow fairly easily from the characterization of containment for conjunctive queries and unions thereof. We show that LAV simplification is NP-complete in the case of both sound and exact schema mappings based on conjunctive queries. In the case of unions of conjunctive queries, the problem is still in NP for sound mappings, while it is in Π_2^p for exact ones.

For graph database schema mappings based on regular path queries, we prove that LAV simplification is PSPACE-complete under the sound semantics, and in EXPSPACE in the case of exact schema mappings. By exploiting a language-theoretic characterization for containment of regular path queries with inverse (called two-way regular path queries) provided in [22], we also extend the results to the case where queries in schema mappings are two-way regular path queries, as well as conjunctive two-way regular path queries, and unions of such queries.

Note that a regular path query returns the set of node pairs in the graph database connected by a path conforming to the query, and therefore can be seen as the regular language constituted by all the words labeling the paths denoted by the query. Indeed, the simplification problem addressed in this paper has a language theoretic interpretation in terms of language equations. Specifically, we address systems of language constraints of the form

$$e_1 \subseteq e_2, \quad \text{and} \quad e_1 = e_2.$$

Here, while e_2 is an ordinary regular expression, e_1 is a regular expression that may contain variables as additional alphabet symbols, to be substituted by an ordinary regular expression. The key idea of our approach is that we can prove that solutions of the above equations are closed under congruence, which enables us to represent languages as graphs over the finite-state automaton for e_2.

The paper is organized as follows. After a discussion on related work in Section 2, we recall some preliminary notions in Section 3. In Section 4, we formally define the problem of schema mapping simplification based on logical implication. In Section 5, we study the problem in the case where queries and views are conjunctive queries, and unions thereof, including a brief discussion on GAV simplification. In Section 6 we illustrate the techniques for the case of regular path queries over graph databases. In Section 7 we extend them to two-way regular path queries, and discuss the case of (unions of) conjunctions thereof. Finally, Section 8 concludes the paper.

2. Related work

As we said in the Introduction, the issue of representing schema mappings, and reasoning on them has been widely investigated in the last years. In [23–27] the emphasis is on providing foundations for data exchange systems based on schema mappings. Other works deal with answering queries posed to the target schema on the basis of both the data at the sources, and a set of source-to-target mapping assertions (see, for instance, [28,25,29] and the surveys in [6,30,2]). A large body of work has been devoted to studying operators on schema mappings relevant to model management, notably, composition, merge, and inverse (see, for example [31–34,11,35,36,12,37]).

Recently, there has been a growing interest in principles and tools for comparing both schema mapping languages, and schema mappings expressed in a certain language. Comparing schema mapping languages aims at characterizing such languages in terms of both expressive power, and complexity of mapping-based computational tasks [7,8]. In particular, [7] studies various relational schema mapping languages with the goal of characterizing them in terms of structural properties possessed by the schema mappings specified in these languages.

Methods for comparing schema mappings have been recently proposed in [9–12]. In [11,12], schema mappings are compared with respect to their ability to transfer source data and avoid redundancy in the target databases, as well as their ability to cover target data. More relevant to the present paper is the work in [9], which introduces three notions of equivalence. The first one is the usual notion based on logic: two schema mappings are logically equivalent if they are indistinguishable by the semantics, i.e., if they are satisfied by the same set of database pairs. The other two notions, called data exchange and conjunctive query equivalence, respectively, are relaxations of logical equivalence, capturing indistinguishability for different purposes.

The above discussion shows that the work on optimization and simplification of schema mappings has concentrated so far on equivalence preserving transformations. As we said in the Introduction, we follow a different approach, and address the issue of mapping simplification based on logical implication of schema mappings, rather than logical equivalence.

As we have already observed, the simplification problem in the context of graph databases has a language theoretic interpretation in terms of language equations. In general, solving systems of equations of the form $e = e'$, where e and e' are regular expressions over an alphabet of constants and variables is undecidable, because it is easy to express the universality problem for Context Free Grammars in this way. In [38], the authors study linear equations of the form

$$e_0 + e_1 \cdot x_1 + \cdots + e_n \cdot x_n = e'_0 + e'_1 \cdot x_1 + \cdots + e'_n \cdot x_n$$

where $e_0, e'_0, \ldots, e_n, e'_n$ are regular expressions, and $x_1, \ldots, x_n$ are variables. The authors prove that solving these equations is ExpTime-complete. In contrast, we prove the solvability of another class of problems, namely, systems of language constraints of the form $e_1 \sqsubseteq e_2$ and $e_1 = e_2$ where both e_1 and e_2 are regular expressions, and e_1 contains variables, whereas e_2 has no variables.

Taking into account the language-theoretic view, our work has also connections with [39,40], which also study language constraints of the forms $e_1 \sqsubseteq e_2$, and $e_1 = e_2$. However, in these works, e_1 is restricted to be a single word on both the source and the target alphabets.

3. Preliminaries

In this work we deal with two data models, the standard relational model [41], and the graph database model [42].

Given a (relational) alphabet Σ, a database over Σ is a finite structure over Σ. We denote a query of arity n over Σ as $\{(x_1, \ldots, x_n) \mid \alpha\}$, where $x_1, \ldots, x_n$ are pairwise distinct variables, called the *distinguished variables* of the query, and α is an expression over Σ containing $x_1, \ldots, x_n$, to be evaluated on databases over Σ in order to compute the assignment to such variables. For a query q over Σ, we denote by $q^{\mathcal{D}}$ the set of tuples resulting from evaluating q in a database $\mathcal{D}$ over Σ. A query q over Σ is *empty* if for each database $\mathcal{D}$ over Σ we have $q^{\mathcal{D}} = \emptyset$. Given two queries q_1 and q_2 over Σ, we say that q_1 is *contained in* q_2, denoted $q_1 \sqsubseteq q_2$, if $q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}}$ for every database $\mathcal{D}$ over Σ. The queries q_1 and q_2 are *equivalent*, denoted $q_1 \equiv q_2$, if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

We assume familiarity with (unions of) conjunctive queries, (U)CQs, over relational databases. In particular, a CQ over Σ is an expression of the form $\{(x_1, \ldots, x_n) \mid \alpha\}$, where α is a conjunction of atoms over Σ that may include constants, distinguished variables, and additional variables that are implicitly existentially quantified. We consider such queries as interpreted under the active domain semantics [41] modified, for technical reasons (see Definition 4.3), by assuming that the active domain contains an additional special constant that does not appear in any relation. For every arity n, we consider two special queries: the *universal query* denoted true/n, returning the n-cartesian product of the active domain, and the *empty query* denoted false/n, returning the empty set. We may omit n when it is clear from the context. We recall that containment between (U)CQs can be characterized in terms of homomorphisms (also called *containment mappings*) [43]: For two CQs q_1 and q_2, we have that $q_1 \sqsubseteq q_2$ iff there is a homomorphism from q_2 to q_1, i.e., a mapping h from the variables and constants of q_2 to those of q_1 that is the identity on distinguished variables and constants and such that, if $r(x_1, \ldots, x_k)$ is an atom of q_2, then $r(h(x_1), \ldots, h(x_k))$ is an atom of q_1. For two UCQ q_1 and q_2, we have that $q_1 \sqsubseteq q_2$ iff for each CQ q_1^i in q_1 there is a CQ q_2^j in q_2 such that $q_1^i \sqsubseteq q_2^j$ [44].

We recall the basic notions regarding graph databases and regular path queries. A *graph database* is a finite graph whose nodes represent objects and whose edges are labeled by elements from an alphabet of binary relational symbols [16,45,46,22]. An edge (o_1, r, o_2) from object o_1 to object o_2 labeled by r represents the fact that relation r holds between o_1 and o_2.

A *regular-path query* (RPQ) over an alphabet Σ of binary relation symbols is a binary query characterized by a regular language. We denote an RPQ $\{(x, y) \mid e(x, y)\}$, where e is a regular expression or a *nondeterministic finite state automaton* (*1NFA*) over Σ, simply by e. When evaluated on a graph database $\mathcal{D}$ over Σ, an RPQ q computes the set $q^{\mathcal{D}}$ of pairs of objects connected in $\mathcal{D}$ by a path in the regular language $\mathcal{L}(q)$ defined by q. We consider also *two-way regular-path queries* (2RPQs) [47,22], which extend RPQs with the *inverse* operator. Formally, let $\Sigma^{\pm} = \Sigma \cup \{r^- \mid r \in \Sigma\}$ be the alphabet including a new symbol r^- for each r in Σ. Intuitively, r^- denotes the inverse of the binary relation r. If $p \in \Sigma^{\pm}$, then we use p^- to mean the *inverse* of p, i.e., if p is r, then p^- is r^-, and if p is r^-, then p^- is r. 2RPQs are expressed by means of a 1NFA over $\Sigma^\pm$. When evaluated on a database $\mathcal{D}$ over Σ, a 2RPQ q computes the set $q^\mathcal{D}$ of pairs of objects connected in $\mathcal{D}$ by a semipath that conforms to the regular language $\mathcal{L}(q)$. A *semipath* in $\mathcal{D}$ from x to y (labeled with $p_1 \cdots p_n$) is a sequence of the form $(y_0, p_1, y_1, \ldots, y_{n-1}, p_n, y_n)$, where $n \geqslant 0$, $y_0 = x$, $y_n = y$, and for each y_{i-1}, p_i, y_i, we have that $p_i \in \Sigma^\pm$, and, if $p_i = r$ then $(y_{i-1}, y_i) \in r^\mathcal{D}$, and if $p_i = r^-$ then $(y_i, y_{i-1}) \in r^\mathcal{D}$. We say that a semipath $(y_0, p_1, \ldots, p_n, y_n)$ *conforms to* q if $p_1 \cdots p_n \in \mathcal{L}(q)$.

We conclude by observing that (U)CQs, RPQs, 2RPQs, and (U)C2RPQs are *monotone*, where a query q is monotone if, whenever $\mathcal{D}_1 \subseteq \mathcal{D}_2$ (i.e., $r^{\mathcal{D}_1} \subseteq r^{\mathcal{D}_2}$ for each relation r) we have that $q^{\mathcal{D}_1} \subseteq q^{\mathcal{D}_2}$.

4. Schema mapping simplification

We refer to a scenario with one source schema, one target schema, and a schema mapping between the two. To model the source and the target schemas we refer to two finite alphabets, the *source alphabet* Σ_s and the *target alphabet* Σ_t, and to specify the mapping, we use a correspondence between queries expressed in a given query language.

Definition 4.1. Given a query language $\mathcal{Q}$, a *$\mathcal{Q}$-based schema mapping assertion from Σ_s to Σ_t* is a statement of the form $q_s \rightsquigarrow q_t$, where q_s and q_t are two queries in $\mathcal{Q}$ with the same arity, respectively over Σ_s and over Σ_t. A *$\mathcal{Q}$-based schema mapping from Σ_s to Σ_t* is a set of mapping assertions from Σ_s to Σ_t.

In the following, we specify explicitly $\mathcal{Q}$, Σ_s, and Σ_t only when they are required or not clear from the context.

We consider two types of schema mappings, called *sound* and *exact*. Intuitively, in a sound mapping, the correspondence between the tuples computed by q_s and those computed by q_t is set containment, while in an exact mapping the correspondence is set equality. Formally, given a source database $\mathcal{D}_s$ and a target database $\mathcal{D}_t$, we say that a sound mapping M is *satisfied* by $(\mathcal{D}_s, \mathcal{D}_t)$, denoted $(\mathcal{D}_s, \mathcal{D}_t) \models M$, if for each mapping assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s^{\mathcal{D}_s} \subseteq q_t^{\mathcal{D}_t}$. Similarly, an exact mapping M is *satisfied* by $(\mathcal{D}_s, \mathcal{D}_t)$ if for each $q_s \rightsquigarrow q_t$ in M, we have that $q_s^{\mathcal{D}_s} = q_t^{\mathcal{D}_t}$.

A fundamental notion in our setting is that of logical implication between mappings.

Definition 4.2. A mapping M_1 *logically implies* a mapping M_2, denoted $M_1 \models M_2$, if for every pair $(\mathcal{D}_s, \mathcal{D}_t)$ such that $(\mathcal{D}_s, \mathcal{D}_t) \models M_1$, we also have that $(\mathcal{D}_s, \mathcal{D}_t) \models M_2$.

We consider two simplified forms of mappings called LAV (local-as-view) and GAV (global-as-view), respectively. In a LAV *assertion* $q_s \rightsquigarrow q_t$, the query q_s is constituted simply by an atom whose predicate symbol belongs to Σ_s and whose arguments are pairwise distinct distinguished variables,[1] while q_t is an arbitrary query.[2] Conversely, in a GAV *assertion*, q_s is an arbitrary query while q_t is constituted simply by an atom whose predicate symbol belongs to Σ_t and whose arguments are pairwise distinct distinguished variables. A LAV *mapping* is a set of LAV assertions with one assertion for each symbol in Σ_s. If M_L is a LAV mapping and $a \in \Sigma_s$, we denote by $M_L(a)$ the target query to which a is mapped by M_L. Conversely, a GAV *mapping* is a set of GAV assertions with one assertion for each symbol in Σ_t. Analogously to the case of LAV mappings, $M_G(a)$ denotes the source query to which the symbol $a \in \Sigma_t$ is mapped by the GAV mapping M_G. Note that some of the queries in a LAV (resp., GAV) mapping may be the empty query.

The problem we consider aims at checking whether a simplified mapping exists that logically implies a given mapping M. Let us consider the case where M is constituted by a single assertion $q_s \rightsquigarrow q_t$, and we aim at simplifying M by means of a LAV schema mapping M'. One trivial way to do this is to define M' in such a way that, whenever a pair $(\mathcal{D}_s, \mathcal{D}_t)$ satisfies M', the query q_s is empty when evaluated over $\mathcal{D}_s$. Similarly, a trivial GAV simplification M' would be one that enforces that the query q_t returns the whole active domain, when evaluated over $\mathcal{D}_t$. We would like to rule out these meaningless simplifications, and therefore we formally introduce the notion of *triviality* w.r.t. a mapping assertion, distinguishing the two cases of LAV and GAV.

Definition 4.3. A LAV (resp., GAV) mapping M' is said to be *trivial* w.r.t. a mapping assertion $q_s \rightsquigarrow q_t$, if for each pair $(\mathcal{D}_s, \mathcal{D}_t)$ with $(\mathcal{D}_s, \mathcal{D}_t) \models M'$, we have that $q_s^{\mathcal{D}_s} = \mathsf{false}^{\mathcal{D}_s}$ (resp., $q_t^{\mathcal{D}_t} = \mathsf{true}^{\mathcal{D}_t}$). A LAV or GAV mapping M' is said to be *trivial* w.r.t. a mapping M, denoted $M' \models_{triv} M$, if there is a mapping assertion $q_s \rightsquigarrow q_t \in M$ such that M' is trivial w.r.t. $q_s \rightsquigarrow q_t$.

We observe that our assumption of having an additional constant in the active domain plays a role here. Indeed, without such an assumption, if the target signature consists of a single unary predicate, then the denotation of that predicate in a target instance would necessarily coincide with the active domain of that instance, and therefore, by definition, $M' \models_{triv} M$ would hold, regardless of what M and M' look like.

[1] Note that the assumption of having pairwise distinct variables is in line with the data integration framework, where all source tuples should be mapped to the target.

[2] For RPQs and 2RPQs, where query variables are not represented explicitly, we consider an atom to be simply a binary predicate symbol.

As an example of Definition 4.3, consider a mapping M constituted by the two assertions

$$\{(x) \mid a_1(x) \wedge a_2(x)\} \rightsquigarrow \{(x) \mid b_1(x) \wedge b_2(x)\},$$
$$\{(x) \mid a_1(x)\} \rightsquigarrow \{(x) \mid b_3(x)\}.$$

Then, the following LAV mapping M' is trivial w.r.t. M, since for every pair $(\mathcal{D}_s, \mathcal{D}_t)$ satisfying M', we have that $\{(x) \mid a_1(x) \wedge a_2(x)\}$ evaluates to $\emptyset$ over $\mathcal{D}_s$:

$$\{(x) \mid a_1(x)\} \rightsquigarrow \mathsf{true}/1,$$
$$\{(x) \mid a_2(x)\} \rightsquigarrow \mathsf{false}/1.$$

Definition 4.4. Let *tp* be one of SOUND or EXACT, *fm* one of LAV or GAV, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ two query languages. *Mapping simplification*, denoted

$$\text{MSIMP}[tp, fm, \mathcal{Q}_1, \mathcal{Q}_2],$$

is the following decision problem: given a $\mathcal{Q}_1$-based schema mapping M of type *tp*, check whether there exists a $\mathcal{Q}_2$-based schema mapping M' of type *tp* and form *fm* such that $M' \models M$, and $M' \not\models_{triv} M$.

To rule out (uninteresting) cases where all mappings that imply a given mapping M are trivial w.r.t. M, in the following we require that for each mapping assertion $q_s \rightsquigarrow q_t$ in a LAV mapping M, q_t is different from false. Analogously, we require that for each mapping assertion $q_s \rightsquigarrow q_t$ in a GAV mapping M, q_s is different from true.

If a simplified mapping exists, it is also of interest to actually compute one. Therefore, we consider the corresponding synthesis problem.

Definition 4.5. Let *tp* be one of SOUND or EXACT, *fm* one of LAV or GAV, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ two query languages. *Mapping synthesis*, denoted,

$$\text{MSYNT}[tp, fm, \mathcal{Q}_1, \mathcal{Q}_2],$$

is the following problem: given a $\mathcal{Q}_1$-based schema mapping M of type *tp*, find a $\mathcal{Q}_2$-based schema mapping M' of type *tp* and form *fm* such that $M' \models M$, and $M' \not\models_{triv} M$.

In general we may want to synthesize simplified mappings with specific interesting properties. As an example, in this paper we are interested in the tightest simplifications of a mapping M, i.e., the simplifications that best approximate M.

Definition 4.6. Let *tp* be one of SOUND or EXACT, *fm* one of LAV or GAV, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ two query languages. *Maximal mapping synthesis*, denoted,

$$\text{MAXMSYNT}[tp, fm, \mathcal{Q}_1, \mathcal{Q}_2],$$

is the following problem: given a $\mathcal{Q}_1$-based schema mapping M of type *tp*, find a solution M' of MSYNT$[tp, fm, \mathcal{Q}_1, \mathcal{Q}_2]$ such that no solution M'' of MSYNT$[tp, fm, \mathcal{Q}_1, \mathcal{Q}_2]$ satisfies $M' \models M''$ and $M'' \not\models M'$.

Observe that, in general, we might have more than one maximal solution for the mapping synthesis problem. In the LAV case, a maximal solution M' is such that there is no solution M'' such that: (1) $M'(a) \sqsubseteq M''(a)$ for every source symbol a, and (2) it is not the case that $M''(b) \sqsubseteq M'(b)$ for every source symbol b, i.e., the queries $M'(a)$ are maximal. Conversely, in the GAV case, a maximal solution M' is such that the queries $M'(b)$ are minimal, for each target symbol b.

In this paper we study the above problems for a variety of cases, where $\mathcal{Q}_1$ and $\mathcal{Q}_2$ range over (U)CQs and variants of queries over graph databases.

We start by observing that we can characterize mapping implication, and hence mapping simplification, in terms of query unfolding w.r.t. a set of mappings. We make use of such a characterization in the technical development in the subsequent sections. The notion of query unfolding is formally defined as follows: let q_s be a source query and M_L a LAV mapping. The *unfolding of* q_s *w.r.t.* M_L, denoted $q_s[M_L]$, is the target query obtained by replacing each atom α in q_s whose predicate symbol is a with $M_L(a)$. An analogous definition holds for the unfolding $q_t[M_G]$ of a target query q_t w.r.t. a GAV mapping M_G.

Proposition 4.1. *Let $\mathcal{Q}$ be a monotone query language.*

(1) *Let M_L be a* LAV *mapping and M a mapping, both $\mathcal{Q}$-based and of type* SOUND *(resp.,* EXACT*). Then $M_L \models M$ iff for each assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s[M_L] \sqsubseteq q_t$ (resp., $q_s[M_L] \equiv q_t$).*

(2) *Let M_G be a* GAV *mapping and M a mapping, both $\mathcal{Q}$-based and of type* SOUND *(resp.,* EXACT*). Then $M_G \models M$ iff for each assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s \sqsubseteq q_t[M_G]$ (resp., $q_s \equiv q_t[M_G]$).*

Proof. We provide the proof for (1) for the case of sound mappings. The other cases can be proved analogously. We start with the following observation: if M_L is a LAV mapping, and $\mathcal{D}_s$ is the source database obtained from a target database $\mathcal{D}_t$ by letting $r^{\mathcal{D}_s} = M_L(r)^{\mathcal{D}_t}$ for every $r \in \Sigma_s$, then for every source query q, we have that $q^{\mathcal{D}_s} = q[M_L]^{\mathcal{D}_t}$.

"⇒" Now, assume that there is an assertion $q_s \rightsquigarrow q_t$ in M such that $q_s[M_L] \not\sqsubseteq q_t$, and let $\mathcal{D}_t$ be such that for some tuple $\mathbf{d}$ we have $\mathbf{d} \in q_s[M_L]^{\mathcal{D}_t}$, and $\mathbf{d} \notin q_t^{\mathcal{D}_t}$. Let $\mathcal{D}_s$ be the source database obtained from $\mathcal{D}_t$ by letting $r^{\mathcal{D}_s} = M_L(r)^{\mathcal{D}_t}$ for every $r \in \Sigma_s$. Clearly, $(\mathcal{D}_s, \mathcal{D}_t) \models M_L$. By the above observation, we have that $q_s^{\mathcal{D}_s} = q_s[M_L]^{\mathcal{D}_t}$, and therefore $\mathbf{d} \in q_s^{\mathcal{D}_s}$. It follows that $(\mathcal{D}_s, \mathcal{D}_t) \not\models q_s \rightsquigarrow q_t$, and $M_L \not\models M$. A contradiction.

"⇐" Assume that $M_L \not\models M$, i.e., there are an assertion $q_s \rightsquigarrow q_t$ in M, and a pair $(\mathcal{D}_s, \mathcal{D}_t)$ such that $(\mathcal{D}_s, \mathcal{D}_t) \models M_L$, and $q_s^{\mathcal{D}_s} \not\subseteq q_t^{\mathcal{D}_t}$, which means that there is $\mathbf{d}$ such that $\mathbf{d} \in q_s^{\mathcal{D}_s}$ but $\mathbf{d} \notin q_t^{\mathcal{D}_t}$. Let $\mathcal{D}'_s$ be such that $r^{\mathcal{D}'_s} = M_L(r)^{\mathcal{D}_t}$ for every $r \in \Sigma_s$. Clearly, $\mathcal{D}'_s \supseteq \mathcal{D}_s$, and $(\mathcal{D}'_s, \mathcal{D}_t) \models M_L$. Since q_s is monotone, we have that $\mathbf{d} \in q_s^{\mathcal{D}'_s} = q_s[M_L]^{\mathcal{D}_t}$, and therefore $q_s[M_L] \not\sqsubseteq q_t$. A contradiction. □

Proposition 4.2. *Let $\mathcal{Q}$ be a monotone query language, and let M be a mapping and M' a* LAV *(resp.,* GAV*) mapping, both $\mathcal{Q}$-based and of type* SOUND *or* EXACT*. Then $M' \models_{triv} M$ iff for some mapping assertion $q_s \rightsquigarrow q_t \in M$ we have that $q_s[M'] \sqsubseteq \mathsf{false}$ (resp., $\mathsf{true} \sqsubseteq q_t[M']$).*

Proof. We first provide the proof for the case of (SOUND or EXACT) LAV mappings.

"⇒' ' Since $M' \models_{triv} M$, there is a mapping assertion $q_s \rightsquigarrow q_t \in M$ such that for each pair $(\mathcal{D}_s, \mathcal{D}_t)$ with $(\mathcal{D}_s, \mathcal{D}_t) \models M'$, we have that $q_s^{\mathcal{D}_s} = \mathsf{false}^{\mathcal{D}_s}$. Assume that $q_s[M'] \not\sqsubseteq \mathsf{false}$, i.e., that $q_s[M']^{\mathcal{D}'_t} \neq \emptyset$ for some target database $\mathcal{D}'_t$. But then, for the source database $\mathcal{D}'_s$ obtained from $\mathcal{D}'_t$ by letting $r^{\mathcal{D}'_s} = M'(r)^{\mathcal{D}'_t}$ for every $r \in \Sigma_s$, we have that $(\mathcal{D}'_s, \mathcal{D}'_t) \models M'$ (both when M' is of type SOUND and of type EXACT) and $q_s^{\mathcal{D}'_s} \neq \mathsf{false}^{\mathcal{D}'_s}$. A contradiction.

"⇐" Assume that for some mapping assertion $q_s \rightsquigarrow q_t \in M$ we have that $q_s[M'] \sqsubseteq \mathsf{false}$ and that $M' \not\models_{triv} M$. Then there exists a pair $(\mathcal{D}'_s, \mathcal{D}'_t)$ such that $(\mathcal{D}'_s, \mathcal{D}'_t) \models M'$ and $q_s^{\mathcal{D}'_s} \neq \mathsf{false}^{\mathcal{D}'_s}$, i.e., $q_s^{\mathcal{D}'_s} \neq \emptyset$. Consider the source database $\mathcal{D}''_s$ obtained from $\mathcal{D}'_t$ by letting $r^{\mathcal{D}''_s} = M'(r)^{\mathcal{D}'_t}$ for every $r \in \Sigma_s$. Since $(\mathcal{D}'_s, \mathcal{D}'_t) \models M'$, we have that $r^{\mathcal{D}'_s} \subseteq M'(r)^{\mathcal{D}'_t}$ (actually, $r^{\mathcal{D}'_s} = M'(r)^{\mathcal{D}'_t}$ when M' is EXACT), hence by monotonicity of q_s, we have that $q_s^{\mathcal{D}'_s} \subseteq q_s^{\mathcal{D}''_s} = q_s[M']^{\mathcal{D}'_t} = \mathsf{false}^{\mathcal{D}'_t} = \emptyset$. A contradiction.

As for the case of GAV mappings, the proof is analogous, but makes use of the fact that the active domain contains an extra constant to deal with the special case where the target alphabet consists of a single unary relation. □

For many of the results in the next sections, we make use of the above characterization, without further mentioning Propositions 4.1 and 4.2.

5. Simplification and synthesis for (U)CQs

In this section, we consider the case of mappings based on conjunctive queries (CQs) and their unions (UCQs), and study the problem of simplifying a given mapping M in terms of a LAV or a GAV mapping. The techniques we adopt for establishing our upper bounds are based on determining a polynomial bound on the length of the queries to consider when searching for the simplified mapping logically implying M, and are reminiscent of those in [48].

5.1. The case of LAV

In the following, when we refer to a LAV mapping logically implying a given mapping, we implicitly assume that implication is non-trivial. We start with the problem of simplifying a CQ-based mapping in terms of a CQ-based LAV mapping.

Theorem 5.1. *Both* MSIMP[SOUND,LAV,CQ,CQ] *and* MSIMP[EXACT,LAV,CQ,CQ] *are in NP.*

Proof. Consider a sound CQ-based mapping consisting of a single assertion $q_s \rightsquigarrow q_t$, where q_t contains ℓ_{q_t} atoms, and a sound CQ-based LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$. Then, there exists a homomorphism from q_t to $q_s[M_L]$, and at most ℓ_{q_t} atoms of $q_s[M_L]$ are in the image of this homomorphism. Hence, for each symbol $a \in \Sigma_s$ occurring in q_s, only at most ℓ_{q_t} atoms in query $M_L(a)$ are needed for the homomorphism. In the general case where the mapping M consists of several assertions, for each $a \in \Sigma_s$ we need at most $\ell_M = \sum_{q_s \rightsquigarrow q_t \in M} \ell_{q_t}$ atoms in the query $M_L(a)$, in order to guarantee the existence of the homomorphisms for all the assertions in M. Hence, in order to check for the existence of an appropriate LAV mapping M_L, it suffices to guess, for each symbol $a \in \Sigma_s$ appearing in one of the mapping assertions in M, a CQ $M_L(a)$ over Σ_t of size at most ℓ_M, and check that $q_s[M_L] \sqsubseteq q_t$, for each $q_s \rightsquigarrow q_t \in M$. In doing so, we rule out the guess of mappings that are trivial w.r.t. M. This gives us immediately an NP upper bound for MSIMP[SOUND,LAV,CQ,CQ].

For MSIMP[EXACT,LAV,CQ,CQ], in addition to checking that $q_s[M_L] \sqsubseteq q_t$, we need also to check that $q_t \sqsubseteq q_s[M_L]$. We observe that the bound on the number of atoms in $M_L(a)$ derived for the sound case is still valid, since if $q_t \sqsubseteq q_s[M_L]$ for a LAV mapping M_L, then also $q_t \sqsubseteq q_s[M'_L]$ for every LAV mapping M'_L such that $M'_L(a)$ is constituted by a subset of the atoms of $M_L(a)$. Therefore, the overall complexity does not change. □

For the case where M is UCQ-based, and the LAV mapping M_L is still CQ-based, we can generalize the above argument by considering containment between UCQs instead of containment between CQs.

Theorem 5.2. *Both* MSIMP[SOUND,LAV,UCQ,CQ] *and* MSIMP[EXACT,LAV,UCQ,CQ] *are in NP.*

Proof. Consider a sound UCQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$ and a sound CQ-based LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$. We remind that $q_s[M_L] \sqsubseteq q_t$ if for each CQ q_1 in the UCQ $q_s[M_L]$ there is a CQ q_2 in the UCQ q_t such that $q_1 \sqsubseteq q_2$. Then, for each CQ q_1 in $q_s[M_L]$ there exists a homomorphism from some CQ q_2 in q_t to q_1, and for each $a \in \Sigma_s$ at most ℓ_{q_t} atoms in the query $M_L(a)$ are needed for the homomorphism, where ℓ_{q_t} is the maximum number of atoms among the CQs in q_t. In the general case where the mapping M consists of several assertions, for each $a \in \Sigma_s$ at most $\ell_M = \sum_{q_s \rightsquigarrow q_t \in M} \ell_{q_t}$ atoms in the query $M_L(a)$ are needed for the homomorphisms for all the assertions in M. Thus, in order to check for the existence of an appropriate LAV mapping M_L, it suffices to guess (again avoiding trivial mappings), for each symbol $a \in \Sigma_s$ appearing in M, a CQ $M_L(a)$ over Σ_t of size at most $\ell_M = \sum_{q_s \rightsquigarrow q_t \in M} \ell_{q_t}$, and check for each $q_s \rightsquigarrow q_t \in M$ that $q_s[M_L] \sqsubseteq q_t$ (and $q_t \sqsubseteq q_s[M_L]$ for the exact variant). □

The last case we consider is the one where both M and the LAV mapping M_L are UCQ-based. In the sound case, we show that simplification to a UCQ-based LAV mapping is equivalent to simplification to a CQ-based LAV mapping.

Lemma 5.3. MSIMP[SOUND,LAV,UCQ,UCQ] *admits a solution for a mapping M iff* MSIMP[SOUND,LAV,UCQ,CQ] *admits a solution for M.*

Proof. Let M be a sound UCQ-based mapping and M_L a sound UCQ-based LAV mapping such that $M_L \models M$. Consider the CQ-based LAV mapping M'_L obtained from M_L by choosing, for each symbol a in Σ_s, as $M'_L(a)$ one of the CQs in $M_L(a)$. We show that $M'_L \not\models_{triv} M$, and that $M'_L \models M$. Consider one assertion $q_s \rightsquigarrow q_t \in M$ such that $q_s[M_L]$ is a non-empty positive query. Such a mapping assertion exists, since $M_L \not\models_{triv} M$. Then, $q_s[M'_L]$ is a non-empty UCQ, and hence $M'_L \not\models_{triv} M$. To show that $M'_L \models M$, it is sufficient to observe that, for each assertion $q_s \rightsquigarrow q_t \in M$, each CQ in $q_s[M'_L]$ is contained in $q_s[M_L]$, and hence in q_t. □

By the above lemma, we trivially get:

Theorem 5.4. MSIMP[SOUND,LAV,UCQ,UCQ] *is in NP.*

For the exact case, the analog of Lemma 5.3 does not hold. In this case we are able to show a higher upper bound.

Theorem 5.5. MSIMP[EXACT,LAV,UCQ,UCQ] *is in* Π_2^p.

Proof. Consider an exact UCQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$ and an exact UCQ-based LAV mapping M_L such that $M_L \models M$. Let m_{q_t} be the number of CQs in q_t, and ℓ_{q_t} the number of atoms in the longest CQ in q_t. Let q'_{s,M_L} be the UCQ obtained from $q_s[M_L]$ by distributing, for each atom α of q_s, the unions in the UCQ $\alpha[M_L]$ over the conjunctions of each CQ of q_s. Since $q_s[M_L] \sqsubseteq q_t$, for each CQ in q'_{s,M_L}, there is a homomorphism from some CQ in q_t to it. Hence, for each symbol $a \in \Sigma_s$ occurring in q_s, for each CQ in the UCQ $M_L(a)$, we need at most ℓ_{q_t} atoms for the homomorphisms from all CQs in q_t. To derive a bound for the number of such CQs, we observe that the inclusion $q_t \sqsubseteq q_s[M_L]$ must also hold. To satisfy this inclusion, it suffices to have in $M_L(a)$ one CQ for each occurrence of a in q_t, i.e., at most $m_{q_t} \cdot \ell_{q_t}$ CQs. It follows that, to check for the existence of the LAV mapping M_L and the corresponding homomorphism, it suffices to guess for each $a \in \Sigma_s$ a UCQ over Σ_t consisting of at most $m_{q_t} \cdot \ell_{q_t}$ CQs, each with at most ℓ_{q_t} atoms. In the general case where the mapping M consists of several assertions $q_s \rightsquigarrow q_t$, we can proceed analogously to the case above, using instead of m_{q_t} and ℓ_{q_t}, the sum of these parameters over all mapping assertions in M. To check whether $q_t \sqsubseteq q_s[M_L]$, for each of the CQs in q_t, it suffices to guess (i) a CQ q' in q_s, (ii) a CQ in $M_L(a)$ for each occurrence of a in q', and (iii) a mapping h from the query q'' obtained by unfolding q' with the selected CQs for each occurrence of a, and check whether h is a homomorphism. This can be done in NP. To check whether $q_s[M_L] \sqsubseteq q_t$, we have to check whether for each CQ q' obtained by selecting one of the CQs q'' in q_s and then substituting each atom α in q'' with one of the CQs in $\alpha[M_L]$, there is a homomorphism from some CQ in q_t to q'. We can do so by a coNP computation that makes use of an NP oracle to check for the existence of a homomorphism. This gives us the Π_2^p upper bound. □

We now show that the upper bounds for the sound cases established in Theorems 5.1, 5.2, and 5.4 are tight.

Theorem 5.6. MSIMP[SOUND,LAV,CQ,CQ] *is NP-hard.*

Proof. The proof is by a reduction from 3-colorability. Given a graph $G = (N, E)$, with $N = \{n_1, \ldots, n_k\}$, we define the corresponding instance of MSIMP[SOUND,LAV,CQ,CQ] as follows.

As for the source alphabet, we consider $\Sigma_s = \{a_e/2, a_s/2, a_f/2\}$, where intuitively a_e is a relation denoting graph edges, a_s is a relation between (starting) elements and graph nodes, and a_t is a relation between graph nodes and (final) elements. Similarly, for the target alphabet, we consider $\Sigma_t = \{b_e/2, b_s/2, b_f/2\}$. On Σ_s and Σ_t we define mapping M as follows:

$$q_T \rightsquigarrow q_G, \tag{1}$$

$$\{(x,y) \mid a_e(x,y)\} \rightsquigarrow \{(x,y) \mid b_e(x,y)\}, \tag{2}$$

$$\{(x,y) \mid a_s(x,y)\} \rightsquigarrow \{(x,y) \mid b_s(x,y)\}, \tag{3}$$

$$\{(x,y) \mid a_f(x,y)\} \rightsquigarrow \{(x,y) \mid b_f(x,y)\} \tag{4}$$

where

$$\begin{aligned} q_T = \{(s,f) \mid\ & a_s(s,r) \wedge a_s(s,g) \wedge a_s(s,b) \wedge \\ & a_e(r,g) \wedge a_e(g,r) \wedge a_e(r,b) \wedge a_e(b,r) \wedge a_e(g,b) \wedge a_e(b,g) \wedge \\ & a_f(r,f) \wedge a_f(g,f) \wedge a_f(b,f)\}, \\ q_G = \Big\{(s,f) \mid\ & b_s(s,x_1) \wedge \cdots \wedge b_s(s,x_k) \wedge \\ & \bigwedge\nolimits_{(n_i,n_j)\in E} \big(b_e(x_i,x_j) \wedge b_e(x_j,x_i)\big) \wedge \\ & b_f(x_1,f) \wedge \cdots \wedge b_f(x_k,f)\Big\}. \end{aligned}$$

Intuitively, assertion (1) maps a triangle, whose three vertexes are connected by a_s and a_f to the distinguished variables s and f respectively, to the graph G, whose nodes are connected by b_s and b_f to the distinguished variables s and f respectively.

We show that G is 3-colorable iff MSimp[sound,lav ,cq,cq] with input M admits a solution. For the "only-if" part, consider the lav mapping M_L consisting of the mapping assertions (2), (3), and (4). If G is 3-colorable, a coloring of the nodes of G with the three colors r, g, b gives us immediately a homomorphism from $q_G[M_L]$ to q_T in which each variable x_i of $q_G[M_L]$ is mapped to the variable of q_T corresponding to the color assigned to node n_i. Hence we have that $q_T \sqsubseteq q_G[M_L]$. For the "if-part", consider a lav mapping M_L such that $q_s[M_L] \sqsubseteq q_t$ for each mapping assertion $q_s \rightsquigarrow q_t$ in M. By the mapping assertions (2), (3), and (4), we have that the queries $M_L(a_e)$, $M_L(a_s)$, and $M_L(a_f)$, which we assume to have (x,y) as distinguished variables, must include respectively the atoms $b_e(x,y)$, $b_s(x,y)$, and $b_f(x,y)$, plus possibly additional atoms containing existentially quantified variables. Note that these existential variables appear in the unfolding $q_T[M_L]$. Now, consider a homomorphism h from $q_G(s,f)$ to $q_T[M_L](s,f)$. Since s and f are distinguished variables, we have that $h(s) = s$ and $h(f) = f$. Suppose that for some variable $x_i \in \{x_1,\dots,x_k\}$ of q_G, we have that $h(x_i)$ is an existential variable y in an additional atom in $q_T[M_L]$. Then, since q_G contains the atoms $b_s(s,x_i)$ and $b_f(x_i,f)$, we must have that $q_T[M_L]$ contains the atoms $b_s(s,y)$ and $b_f(y,f)$. This is impossible, since y is an existential variable introduced by the unfolding of q_T with M_L, and hence can appear in the unfolding of just one atom of q_T. But there is no atom of q_T that contains both s and f, and that could generate both $b_s(s,y)$ and $b_f(y,f)$. So, the only possibility for a homomorphism from $q_G[M_L]$ to q_T is to map each x_i of $q_G[M_L]$ to one of the variables r, g, or b. The existence of such a homomorphism implies that G is 3-colorable. □

Corollary 5.7. MSimp[sound,lav,ucq,cq] *and* MSimp[sound,lav,ucq,ucq] *are NP-hard.*

Proof. From Theorem 5.6 we trivially get the result for MSimp[sound,lav,ucq,cq], and by considering Lemma 5.3, we get the result also for MSimp[sound,lav,ucq,ucq]. □

It is easy to see that the proof of Theorem 5.6 shows also NP-hardness of simplification for exact mappings.

Theorem 5.8. MSimp[exact,lav,cq,cq] *and hence* MSimp[exact,lav,ucq,cq] *are NP-hard.*

We conjecture that the Π_2^p upper bound for MSimp[exact,lav,ucq,ucq] is also tight.

Next, we turn to maximal solutions to lav simplification. We observe that all our upper bound results are based on deriving a bound on the number of atoms that may constitute the queries in a possible solution to the simplification problem, and on the possibility to guess (or enumerate) all solutions within the derived bound. Hence, it becomes possible to check also further properties of the guessed (or enumerated) solutions. Specifically, we can obtain a cq-based lav-mapping that is a maximal solution to simplification of ucq-based mappings by guessing a candidate mapping and checking that it is a solution; then, to check that it is maximal, we generate all other mappings and check that, if they are solutions, they are contained in our candidate solution. This immediately results in a technique to solve maximal mapping synthesis in Σ_3^p.

Theorem 5.9. *A solution to* MAXMSYNT[LAV,SOUND,CQ,CQ] *and to* MAXMSYNT[LAV,SOUND,UCQ,CQ] *can be computed in* Σ_3^p.

Unfortunately, we are not able to use the same argument for deriving a solution to MAXMSYNT[LAV,SOUND,UCQ,UCQ] in Σ_3^p, since we have no guarantee that a maximal UCQ-based solution is of polynomial size.

5.2. The case of GAV

Next, we consider the case of simplifying (U)CQ-based mappings in terms of GAV mappings. Note that for exact mappings, GAV simplification is the same problem as LAV simplification, so we focus here on sound mappings. We start by considering sound CQ-based mappings and show the following upper bound.

Theorem 5.10. MSIMP[SOUND,GAV,CQ,CQ] *is in NP.*

Proof. Consider a sound CQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$ and assume there exists some sound CQ-based GAV mapping M_G such that $q_s \sqsubseteq q_t[M_G]$, witnessed by a homomorphism h, and $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$. We show that there is a sound CQ-based GAV mapping M'_G of bounded size such that $q_s \sqsubseteq q_t[M'_G]$. Indeed, since $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$, there must be one distinguished variable x and one symbol $b \in \Sigma_t$, such that x occurs in an atomic formula with the symbol b in $M_G(b)$. We now obtain $M'_G(b)$ from $M_G(b)$ by selecting in $M_G(b)$ such an atom. For all other symbols $b' \in \Sigma_t$, we take $M'_G(b')$ to be true. By constructing M'_G in this way, we have that the atoms in $q_t[M'_G]$ are a subset of the atoms in $q_t[M_G]$, and hence the projection of h on such atoms is still a homomorphism to q_s.

In the general case where the mapping M consists of k assertions, we can apply the above argument for each of the assertions in M. This shows that, if there exists some sound GAV mapping M_G such that $M_G \models M$, then there is also a GAV mapping M'_G such that $M'_G(b)$ has at most k atoms and $M'_G \models M$. Hence, in order to check the existence of an appropriate GAV mapping M_G, it suffices to guess (avoiding trivial mappings), for each symbol $b \in \Sigma_t$ appearing in M, a CQ $M_G(b)$ over Σ_s of size at most k, and check that $q_s \sqsubseteq q_t[M_G]$, for each $q_s \rightsquigarrow q_t \in M$. □

This result extends immediately to UCQ-based mappings, by checking containment between UCQs, instead of CQs.

Theorem 5.11. MSIMP[SOUND,GAV,UCQ,CQ] *is in NP.*

For UCQ-based mappings, we get the same upper bound, with a somewhat subtler argument.

Theorem 5.12. MSIMP[SOUND,GAV,UCQ,UCQ] *is in NP.*

Proof. We consider first the case of a sound UCQ-based mapping M consisting of a single assertion $q_s \rightsquigarrow q_t$. Assume there exists some sound UCQ-based GAV mapping M_G such that $q_s \sqsubseteq q_t[M_G]$ and $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$. First note that $q_s \sqsubseteq q_t[M_G]$, if for each CQ q_s^i of q_s we have that $q_s^i \sqsubseteq q_t[M_G]$. This means that there are a CQ q_t^i of q_t and a CQ q_b^i for each symbol $b \in \Sigma_t$ such that there is a homomorphism from $q_t^i[M'_G]$ to q_s^i, where $M'_G(b) = q_b^i$. Thus, if q_s is a union of ℓ CQs, then we can assume that each $M_G[b]$ for $b \in \Sigma_t$ has at most ℓ CQs. What is left is to bound the size of these CQs.

Let m be the number of CQs in q_t. Since $q_t[M_G] \neq \mathsf{true}^{\mathcal{D}_t}$, for each CQ q'_t of q_t, there must be one distinguished variable x and one symbol $b \in \Sigma_t$ such that x occurs in some atomic formula with the symbol b in each CQ q'_b of $M_G[b]$. Define M'_G to keep all such atomic formulas, and only such atomic formulas. $M'_G[b]$ contains at most $\ell \cdot m$ atomic formulas. If we have k mapping assertions, then $M'_G[b]$ needs to contain only klm atomic formulas. To check the existence of a simplifying GAV mapping it suffices to guess a mapping M'_G under such a size bound and check that $q_s \sqsubseteq q_t[M'_G]$. □

We conjecture that the above upper bounds are tight.

We observe that the arguments we have used above to derive bounds on the size of solutions to GAV simplification, cannot be used to derive bounds for maximal GAV solutions, i.e., solutions in which the source queries in the mapping assertions are as small as possible. Indeed, the problem of synthesizing maximal (U)CQ-based GAV mappings is still open.

Our results on simplification for (U)CQs are summarized in Table 1.

6. LAV simplification and synthesis for RPQs

In this section, we consider the case of RPQs over graph databases, and study the problem of simplifying an RPQ-based mapping in terms of an RPQ-based LAV mapping. For our results, we exploit a straightforward language theoretic characterization of containment between RPQs. We observe that GAV simplification is wide open for RPQ-based mappings.

Theorem 6.1. *(See [42].) Let q_1, q_2 be two RPQs, and $\mathcal{L}(q_1)$, $\mathcal{L}(q_2)$ the corresponding regular languages. Then $q_1 \sqsubseteq q_2$ iff $\mathcal{L}(q_1) \subseteq \mathcal{L}(q_2)$.*

In the following, we identify an RPQ q over an alphabet Σ with the language over Σ accepted by the regular expression or 1NFA representing q. Considering the language-theoretic characterization above, it follows from Propositions 4.1 and 4.2 that, if M_L is a LAV mapping and M a mapping, both of type SOUND (resp., EXACT), then $M_L \models M$ and $M_L \not\models_{triv} M$ iff for each assertion $q_s \rightsquigarrow q_t$ in M, we have that $q_s[M_L] \subseteq q_t$ (resp., $q_s[M_L] = q_t$) and $q_s[M_L] \neq \emptyset$. Here, the unfolding $q_s[M_L]$ of q_s w.r.t. M_L denotes the *language* over Σ_t obtained from q_s by expanding in each word in q_s each symbol $a \in \Sigma_s$ with the language $M_L(a)$. Obviously, such a language can be represented by a regular expression or a 1NFA of size linear in the product of the sizes of (the representations of) q_s and M_L.

We start by showing that we can characterize mapping implication $M_L \models M$ between a LAV mapping M_L and a mapping M in terms of a single language containment (for sound mappings) or language equality (for exact mappings). For this, we extend the notion of unfolding of q_s w.r.t. M_L to the case where q_s may contain additional symbols w.r.t. those in Σ_s. In particular, the additional symbols are left unchanged by the unfolding.

Proposition 6.2. *Let M be an RPQ-based mapping of type SOUND (resp., EXACT) from Σ_s to Σ_t, and let # be a symbol not in $\Sigma_s \cup \Sigma_t$. Then there are RPQs $q_{M,s}$ over $\Sigma_s \cup \{\#\}$ and $q_{M,t}$ over $\Sigma_t \cup \{\#\}$, both of size linear in M, such that an RPQ-based LAV mapping M_L of type SOUND (resp., EXACT) is a solution to MSYNT[SOUND,LAV ,RPQ,RPQ] (resp., MSYNT[EXACT,LAV,RPQ,RPQ]) with input M iff $q_{M,s}[M_L] \subseteq q_{M,t}$ (resp., $q_{M,s}[M_L] = q_{M,t}$) and $q_{M,s}[M_L] \neq \emptyset$.*

Proof. Let $M = \{q_{1,s} \rightsquigarrow q_{1,t}, \ldots, q_{k,s} \rightsquigarrow q_{k,t}\}$. We set $q_{M,s} = q_{1,s} \cdot \# \cdots \# \cdot q_{k,s}$ and $q_{M,t} = q_{1,t} \cdot \# \cdots \# \cdot q_{k,t}$. Intuitively, the fresh symbol # acts as a separator for the different parts of $q_{M,s}$ and $q_{M,t}$. It is easy to verify that, for every LAV mapping M_L, we have that $q_{i,s}[M_L] \subseteq q_{i,t}$ (resp., $q_{i,s}[M_L] = q_{i,t}$) for $i \in \{1, \ldots, k\}$ iff $q_{M,s}[M_L] \subseteq q_{M,t}$ (resp., $q_{M,s}[M_L] = q_{M,t}$), and that $q_{i,s}[M_L] \neq \emptyset$ for $i \in \{1, \ldots, k\}$ iff $q_{M,s}[M_L] \neq \emptyset$. □

In the following, let Σ_n be a non-empty alphabet of new symbols disjoint from Σ_s and Σ_t, and let $\Sigma'_s = \Sigma_s \cup \Sigma_n$ and $\Sigma'_t = \Sigma_t \cup \Sigma_n$. By Proposition 6.2, the problem MSIMP[SOUND,LAV,RPQ,RPQ] (or MSIMP[EXACT,LAV,RPQ,RPQ]) can be polynomially reduced to the problem of checking, whether for languages q_s over Σ'_s and q_t over Σ'_t there is a LAV mapping M_L such that $q_s[M_L] \subseteq q_t$ (resp., $q_s[M_L] = q_t$) and $q_s[M_L] \neq \emptyset$.

Our technique for mapping simplification exploits a characterization of regular languages by means of congruence classes [49,50,40]. Recall that two words $u, v \in \Sigma^*$ are *congruent* with respect to a language $L \subseteq \Sigma^*$ if, for all words $x, y \in \Sigma^*$ we have that $xuy \in L$ iff $xvy \in L$. Let $A_t = (\Sigma'_t, S_t, s^0_t, \delta_t, F_t)$ be a 1NFA for q_t. Then A_t defines a set of congruence classes partitioning Σ'^*_t. Each congruence class is characterized by a binary relation $G \subseteq S_t \times S_t$ (i.e., a directed graph over S_t), and we define the congruence class associated with G as

$$\mathcal{L}(G) = \bigl\{ w \in \Sigma'^*_t \bigm| \text{for all } s_1, s_2 \in S_t\colon\ s_2 \in \delta_t(s_1, w) \text{ iff } (s_1, s_2) \in G \bigr\}.$$

Intuitively, each word $w \in \mathcal{L}(G)$ connects s_1 to s_2 in A_t, for each pair $(s_1, s_2) \in G$. For a word $w \in \Sigma'^*_t$, we denote by $[w]_{A_t}$ the congruence class to which w belongs.

It follows immediately from the characterization of congruence classes in terms of binary relations over the states of A_t that the set of congruence classes is closed under concatenation. Specifically, for two congruence classes $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$, respectively with associated relations G_1 and G_2, the binary relation associated with $\mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$ is $G_1 \circ G_2$.[3]

We introduce some notation that we use here, and later in this section:

- $\mathcal{G}_t = 2^{S_t \times S_t}$ denotes the set of binary relations associated with the congruence classes for A_t,
- $G^\varepsilon_t = \{(s, s) \mid s \in S_t\}$, and
- $G^b_t = \{(s_1, s_2) \mid s_2 \in \delta_t(s_1, b)\}$, for each $b \in \Sigma'_t$.

Then, for each $G \in \mathcal{G}_t$, we can characterize the congruence class $\mathcal{L}(G)$ associated with G in terms of a DFA.

Lemma 6.3. *(See [49].) The language $\mathcal{L}(G)$ is accepted by the DFA $A_G = (\Sigma'_t, \mathcal{G}_t, G^\varepsilon_t, \delta_{A_t}, F_G)$, where $F_G = \{G\}$ and $\delta_{A_t}(R, b) = R \circ G^b_t$, for each $R \subseteq S_t \times S_t$ and $b \in \Sigma'_t$.*

Notice that, if A_t has m states, then A_G has 2^{m^2} states.

We observe next that we need to allow for the presence of empty queries in the LAV mapping we are looking for. Consider, e.g., $q_s = (a_1 + a_3) \cdot (a_2 + a_3)$ and $q_t = b_1 \cdot b_2$. It is easy to see that for all LAV mappings M_L such that $q_s[M_L] \subseteq q_t$, we have that $M_L(a_3) = \emptyset$. Thus, for any LAV mapping obtained from simplification we have that no nodes are connected through an edge with label a_3. One such LAV mapping M_L is

$$M_L(a_1) = b_1, \qquad M_L(a_2) = b_2, \qquad M_L(a_3) = \emptyset.$$

Observe also that $[b_1]_{A_t} = \{b_1\}$ and $[b_2]_{A_t} = \{b_2\}$, where A_t is the obvious 1NFA for $b_1 \cdot b_2$.

[3] We use $L_1 \cdot L_2$ to denote concatenation between languages, and $G_1 \circ G_2$ to denote composition of binary relations.

6.1. Upper bounds for sound mapping simplification

We first deal with the case of sound mappings, and prove two preliminary results. The first lemma states that w.l.o.g. we can restrict the attention to LAV mappings in which the queries are *singletons*, i.e., queries that are either empty or constituted by a single word.

Lemma 6.4. *Let q_s be an RPQ over Σ'_s, and q_t an RPQ over Σ'_t. If there exists an* RPQ*-based* LAV *mapping M_L such that $q_s[M_L] \subseteq q_t$, then there exists an* RPQ*-based* LAV *mapping M'_L such that $q_s[M'_L] \subseteq q_t$ in which each query is either a single word over Σ_t or empty.*

Proof. If $q_t = \varepsilon$, we can simply set $M'_L(a) = \varepsilon$, for each $a \in \Sigma_s$. Otherwise, since $q_s[M_L] \neq \emptyset$ and $q_s[M_L] \subseteq q_t$, there exist a nonempty word $a_1 \cdots a_k \in q_s$ and a word $w_1 \cdots w_k \in q_s[M_L]$ and hence in q_t, where $w_j \in M_L(a_j)$. To define the new LAV mapping M'_L, we consider each $a \in \Sigma_s$ appearing in $a_1 \cdots a_k$. Notice that a might appear in $a_1 \cdots a_k$ multiple times, and suppose the occurrences of a are $a_{i_1}, \dots, a_{i_\ell}$, corresponding to $w_{i_1}, \dots, w_{i_\ell}$. We chose arbitrarily one w_{i_j} and set $M'_L(a) = w_{i_j}$. Instead, for each $a \in \Sigma_s$ not appearing in $a_1 \cdots a_k$, we set $M'_L(a) = \emptyset$. Now, $q_s[M'_L] \neq \emptyset$ by construction, and since $M'_L(a) \subseteq M_L(a)$ for every $a \in \Sigma_s$, we have that $q_s[M'_L] \subseteq q_s[M_L] \subseteq q_t$. □

The next lemma shows that one can close queries in LAV mappings under congruence.

Lemma 6.5. *Let q_s be an RPQ over Σ'_s, q_t an RPQ over Σ'_t expressed through a 1NFA A_t, and M_L a singleton mapping such that $q_s[M_L] \subseteq q_t$. Then, for M'_L defined such that*

$$M'_L(a) = \begin{cases} [w^a]_{A_t}, & \text{if } M_L(a) = w^a, \\ \emptyset, & \text{if } M_L(a) = \emptyset, \end{cases}$$

we have that $q_s[M'_L] \subseteq q_t$.

Proof. Let $A_t = (\Sigma_t, S_t, s^0_t, \delta_t, F_t)$. Consider a word $a_1 \cdots a_h \in q_s$. If there is one of the a_i such that $M_L(a_i) = \emptyset$, then $M_L(a_1) \cdots M_L(a_h) = \emptyset \subseteq q_t$. Otherwise, we have that $M_L(a_i) = \{w^{a_i}\}$, for $i \in \{1, \dots, h\}$, and since $w^{a_1} \cdots w^{a_h} \in q_s[M_L] \subseteq q_t$, there is a sequence $s_0, s_1, \dots, s_h$ of states of A_t such that $s_0 = s^0_t$, $s_h \in F_t$, and $s_i \in \delta_t(s_{i-1}, w^{a_i})$, for $i \in \{1, \dots, h\}$. Consider now, for each $i \in \{1, \dots, h\}$, a word $w'_i \in M'_L(a_i) = [w^{a_i}]_{A_t}$. Making use of the characterization of $[w^{a_i}]_{A_t}$ in terms of a binary relation over S_t, we have for each word in $[w^{a_i}]_{A_t}$, and in particular for w'_i, that $s_i \in \delta_t(s_{i-1}, w'_i)$. Hence, $s_h \in \delta_t(s^0_t, w'_1 \cdots w'_h)$ and $w'_1 \cdots w'_h \in q_t$. □

From these two lemmas we get that, when searching for a LAV mapping M_L satisfying $q_s[M_L] \subseteq q_t$, we can restrict the attention to queries that are congruence classes for A_t.

Lemma 6.6. *Let q_s be an RPQ over Σ'_s, and q_t an RPQ over Σ'_t expressed through a 1NFA A_t. If there exists an* RPQ*-based* LAV *mapping M_L such that $q_s[M_L] \subseteq q_t$, then there exists an* RPQ*-based* LAV *mapping M'_L such that $q_s[M'_L] \subseteq q_t$, and such that $M'_L(a)$ is a congruence class for A_t, for each $a \in \Sigma_s$.*

Proof. If there exists an RPQ-based LAV mapping M_L such that $q_s[M_L] \subseteq q_t$, then by Lemma 6.4, w.l.o.g., we can assume that M_L consists of singleton queries. Then, the claim follows from Lemma 6.5. □

We derive now a procedure that, given an RPQ q_s over Σ'_s, and an RPQ q_t over Σ'_t expressed respectively through 1NFAs $A_s = (\Sigma'_s, S_s, s^0_s, \delta_s, F_s)$ and $A_t = (\Sigma'_t, S_t, s^0_t, \delta_t, F_t)$, checks for the existence of a sound RPQ-based LAV mapping M_L such that

(1) $q_s[M_L] \neq \emptyset$, and
(2) $q_s[M_L] \subseteq q_t$.

Specifically, by Lemma 6.6, it is sufficient to consider LAV mappings in which each query is constituted by a single congruence class, which can be represented by a binary relation over the state set S_t of A_t. Hence, for each $a \in \Sigma_s$, we guess such a binary relation G_a and verify that for the LAV mapping M_L defined by $M_L(a) = \mathcal{L}(G_a)$, conditions (1) and (2) are satisfied. In doing so, we exploit Lemma 6.3, which provides a characterization of $\mathcal{L}(G_a)$ in terms of a DFA A_{G_a}.

To check condition (1), we proceed as follows:

1.1 for each $a \in \Sigma_s$, we check whether a is a *bad symbol*, i.e., whether $\mathcal{L}(G_a) = \emptyset$;
1.2 we delete from A_s each transition labeled by a bad symbol; and
1.3 we check whether the resulting 1NFA accepts a non-empty language.

To check condition (2), we proceed as follows:

2.1 we construct a 1NFA A_{M_L} accepting $q_s[M_L]$;
2.2 we construct the 1NFA $A_{\not\subseteq} = A_{M_L} \times \bar{A}_t$ as the product NFA of A_{M_L} and the 1NFA $\bar{A}_t$ accepting $\Sigma_t'^* \setminus q_t$;
2.3 we check $A_{\not\subseteq}$ for emptiness.

To construct A_{M_L}, we observe that a word w is in $q_s[M_L]$ if there are a word $a_1 \cdots a_n \in q_s$, and for $i \in \{1, \dots, n\}$, words $w_i \in \mathcal{L}(G_{a_i})$ such that $w = w_1 \cdots w_n$. Hence, A_{M_L} simulates A_s while accepting words in Σ_t' that are concatenations of words in the various languages $\mathcal{L}(G_{a_i})$. Specifically, $A_{M_L} = (\Sigma_t', S_{M_L}, s^0_{M_L}, \delta_{M_L}, F_{M_L})$, where

- $S_{M_L} = \Sigma_s' \times S_s \times \mathcal{G}_t$;
- $s^0_{M_L} = \Sigma_s' \times \{s_s^0\} \times G_t^\varepsilon$;
- $F_{M_L} = \{(a, s, G_a) \mid s \in F_s, a \in \Sigma_s\} \cup \{(b, s, G_t^b) \mid s \in F_s, b \in \Sigma_n\}$;
- and for each $a \in \Sigma_s'$, $s \in S_s$, $R \in \mathcal{G}_t$, and $b \in \Sigma_t'$,

$$\delta_{M_L}\big((a, s, R), b\big) = \begin{cases} \{(a, s, R \circ G_t^b)\}, & \text{if } a \in \Sigma_s, R \neq G_a; \\ \{(a, s, R \circ G_t^b)\} \cup \bigcup_{a' \in \Sigma_s', s' \in \delta_s(s,a)} \{(a', s', G_t^b)\}, & \text{if } a \in \Sigma_s, R = G_a; \\ \bigcup_{a' \in \Sigma_s', s' \in \delta_s(s,a)} \{(a', s', G_t^\varepsilon)\}, & \text{if } a \in \Sigma_n, a = b; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Theorem 6.7. MSIMP[SOUND,LAV,RPQ,RPQ] *is in* PSPACE.

Proof. By Lemma 6.6, to check whether MSIMP[SOUND,LAV,RPQ,RPQ] admits a solution, it suffices to guess for each symbol $a \in \Sigma_s$ a binary relation G_a over the state set S_t of A_t, and check whether for the resulting RPQ-based LAV mapping M_L conditions (1) and (2) hold. When checking condition (1), the emptiness test in item (1.1) can be done for each $a \in \Sigma_s$ in NLOGSPACE in $|A_{G_a}|$, and since the number of states of A_{G_a} is exponential in $|A_t|$, in PSPACE in $|A_t|$. Knowing the set of bad symbols, the transformation in item (1.2) is linear in $|A_s|$. Finally, the non-emptiness test in item (1.3) can be done in NLOGSPACE in $|A_s|$. When checking condition (2), we do not need to construct A_{M_L}, $\bar{A}_t$, and $A_{\not\subseteq}$ explicitly, but can check the nonemptiness of $A_{\not\subseteq}$ on the fly while constructing A_{M_L} and complementing A_t. Hence, since the number of states of A_{M_L} is linear in $|A_s|$ and exponential in $|A_t|$, we get that condition (2) can be checked in PSPACE in $|A_t|$ and in NLOGSPACE in $|A_s|$. □

6.2. Upper bounds for exact mapping simplification and for maximal mapping synthesis

The method based on congruence classes can be adapted to address also LAV simplification for exact mappings. The difference w.r.t. sound mappings is that in this case we need to consider also LAV mappings in which the queries are unions of congruence classes. Indeed, congruence classes (and hence solutions to the LAV mapping synthesis problem) are not closed under union, as shown by the following example.

Let $q_s = a_1 \cdot a_2$ and $q_t = 00 + 01 + 10$. Then the following two incomparable mappings are solutions to MAXM-SYNT[SOUND,LAV,RPQ,RPQ] when the input mapping is $\{q_s \rightsquigarrow q_t\}$:

$$\begin{aligned} M_L^1(a_1) &= 0, & M_L^2(a_1) &= 0 + 1, \\ M_L^1(a_2) &= 0 + 1, & M_L^2(a_2) &= 0. \end{aligned}$$

Notice that the mapping M_L, where $M_L(a_i) = M_L^1(a_i) + M_L^2(a_i)$, for $i \in \{1, 2\}$, is not a solution, since $q_s[M_L]$ includes 11.

On the other hand, we can show that considering mapping in which the queries are unions of congruence classes is sufficient to obtain maximal unfoldings. We first generalize Lemma 6.5 to non-singleton queries.

Lemma 6.8. *Let q_s be an RPQ over Σ_s', q_t an RPQ over Σ_t' expressed through a 1NFA A_t, and M_L a* LAV *mapping such that $q_s[M_L] \subseteq q_t$. Then for M_L' with*

$$M_L'(a) = \begin{cases} \bigcup_{w \in M_L(a)} [w]_{A_t}, & \textit{if } M_L(a) \neq \emptyset, \\ \emptyset, & \textit{if } M_L(a) = \emptyset, \end{cases}$$

we have that $q_s[M_L'] \subseteq q_t$.

Proof. Consider a word $a_1 \cdots a_h \in q_s$. If there is one of the a_i such that $M_L(a_i) = \emptyset$, then $M_L(a_1) \cdots M_L(a_h) = \emptyset \subseteq q_t$. Otherwise, we have that, for $i \in \{1, \dots, h\}$, for some $w^{a_i} \in M_L(a_i)$, the word $w^{a_1} \cdots w^{a_h} \in q_s[M_L] \subseteq \mathcal{L}(A_t)$. We show that, for each $i \in \{1, \dots, h\}$, we also have that $w^{a_1} \cdots w^{a_{i-1}} \cdot w' \cdot w^{a_{i+1}} \cdots w^{a_h} \in \mathcal{L}(A_t)$, for each $w' \in \bigcup_{w \in M_L(a_i)} [w]_{A_t}$. First, since $q_s[M_L] \subseteq q_t$, if $w^{a_1} \cdots w^{a_h} \in q_s[M_L] \subseteq \mathcal{L}(A_t)$, then, for each $w \in M_L(a_i)$, we also have that $w^{a_1} \cdots w^{a_{i-1}} \cdot w \cdot w^{a_{i+1}} \cdots w^{a_h} \in q_s[M_L] \subseteq \mathcal{L}(A_t)$. Then there is a sequence $s_0, s_1, \dots, s_h$ of states of A_t such that $s_0 = s_t^0$, $s_h \in F_t$, $s_j \in \delta_t(s_{j-1}, w^{a_j})$, for

$j \in \{1, \ldots, i-1, i+1, \ldots, h\}$, and $s_i \in \delta_t(s_{i-1}, w)$. Then, by the definition of congruence classes, for each word $w' \in [w]_{A_t}$, we have that $s_i \in \delta_t(s_{i-1}, w'_i)$, and hence $w^{a_1} \cdots w^{a_{i-1}} \cdot w' \cdot w^{a_{i+1}} \cdots w^{a_h} \in \mathcal{L}(A_t)$. □

Lemma 6.9. *Given a mapping $M = \{q_s \rightsquigarrow q_t\}$, where q_t is defined by a 1NFA A_t, every solution M_L to* MaxMSynt[sound ,lav,rpq,rpq] *with input M is such that each query in M_L is a union of congruence classes for A_t.*

Proof. Consider a solution M_L to MaxMSynt[sound,lav,rpq,rpq] with input M, and assume that for some $a \in \Sigma_s$, $M_L(a)$ is not a union of congruence classes for A_t. Then there are some word $w \in M_L(a)$ and some word $w' \in [w]_{A_t}$ such that $w' \notin M_L(a)$. By Lemma 6.8, the mapping M'_L with $M'_L(a) = M_L(a) \cup \{w'\}$ is also a solution to MSynt[sound,lav,rpq,rpq] with input M, thus contradicting the maximality of M_L. □

By observing that every solution to lav simplification for the exact case has to be a maximal lav mapping that implies the given mapping, we get the following upper bound for lav simplification in the exact case.

Theorem 6.10. MSimp[exact,lav,rpq,rpq] *is in* ExpSpace.

Proof. By Lemma 6.9, we can nondeterministically choose mappings M_L in which the queries are unions of congruence classes and then test whether $q_t = q_s[M_L]$. To do so, we build a 1NFA A_{s,M_L} accepting $q_s[M_L]$ as follows. We start by observing that for each union U of congruence classes, we can build the automaton $A_U = (\mathcal{G}_t, s_t, R_\epsilon, \delta_{A_t}, U)$ accepting the words in U, which incidentally, is deterministic. Hence, by substituting each a-transition in the 1NFA A_s for q_s with the 1NFA A_{U_a}, where $M_L(a) = U_a$, we obtain a 1NFA A_{s,M_L}. Note that, even when A_s is deterministic, A_{s,M_L} may be nondeterministic.

To test $q_s[M_L] \subseteq q_t$, we complement A_t, obtaining the 1NFA $\overline{A_t}$, and check the 1NFA $A_{s,M_L} \times \overline{A_t}$ for emptiness. The size of $A_{s,M_L} \times \overline{A_t}$ is polynomial in the size of A_s and exponential in the size of A_t. Checking for emptiness can be done in exponential time, and considering the initial nondeterministic guess, we get a NExpTime upper bound.

To test $q_t \subseteq q_s[M_L]$, we complement A_{s,M_L}, obtaining the 1NFA $\overline{A_{s,M_L}}$, and check $A_t \times \overline{A_{s,M_L}}$ for emptiness. Since A_{s,M_L} is nondeterministic, complementation is exponential. However, we observe again that such a complementation can be done on the fly in ExpSpace, while checking for emptiness and intersecting with A_t. As a consequence, considering the initial nondeteministic guess, MSimp[exact,lav,rpq,rpq] can be decided in NExpSpace, which is equivalent to ExpSpace. □

Note that the proofs of Theorems 6.7 and 6.10 imply that, w.r.t. lav mapping simplification, considering queries that are RPQs (as opposed to general, possibly non-regular, path languages) is not a restriction, since the existence of general lav mappings implies the existence of regular ones. This is also in line with a similar observation holding for the existence of rewritings of RPQs w.r.t. RPQ views [42].

Finally, we observe that using the machinery based on unions of congruence classes, we can also solve the maximal mapping synthesis problem. We guess a mapping and check that it is a solution to mapping synthesis. To check that it is a maximal solution, we generate all other mappings and check that, if they are solutions, they are contained in our candidate solution.

Theorem 6.11. *A solution to* MaxMSynt[sound,lav,rpq,rpq] *and to* MaxMSynt[exact,lav,rpq,rpq] *can be computed in* ExpSpace.

6.3. Lower bounds for mapping simplification

It turns out that the upper bound established for the sound case is tight:

Theorem 6.12. MSimp[sound,lav,rpq,rpq] *is* PSpace*-hard.*

Proof. The proof is by a reduction from the universality problem for REs. Given an RE e over the alphabet $\Sigma_t = \{b_1, \ldots, b_n\}$, let $\Sigma_s = \{a_1, \ldots, a_n\}$, and let M_e be the mapping constituted by the following assertions:

$$\Sigma_s^* \rightsquigarrow e, \tag{5}$$

$$a_1 \rightsquigarrow b_1 \quad \cdots \quad a_n \rightsquigarrow b_n. \tag{6}$$

We show that e is universal iff MSimp[sound,lav,rpq,rpq] with input M_e admits a solution. For the "only-if" part, assume that e is universal and consider the lav mapping M_L consisting of the mapping assertions (6). We have that $\Sigma_s^*[M_L] = \Sigma_t^*$, and since e is universal, also $\Sigma_s^*[M_L] \subseteq e$. For the "if" part, consider a lav mapping M_L such that $q_s[M_L] \subseteq q_t$ and $q_s[M_L] \neq \emptyset$, for each mapping assertion $q_s \rightsquigarrow q_t$ in M_e. By the mapping assertions (6), we have that $M_L(a_i) \neq \emptyset$ (since a_i is the left-hand side of a mapping assertion) and that $M_L(a_i)$ must include b_i, for $i \in \{1, \ldots, n\}$, and hence $\Sigma_s^*[M_L] = \Sigma_t^*$. Since $\Sigma_s^*[M_L] \subseteq e$, we have that e is universal. □

It is easy to see that the above proof shows also PSpace-hardness of simplification for exact mappings.

Corollary 6.13. MSIMP[EXACT,LAV,RPQ,RPQ] *is* PSPACE-*hard.*

We do not have a tight upper bound for MSIMP[EXACT,LAV,RPQ,RPQ]. We do, however, show a tight lower bound for a generalization of MSIMP[EXACT,LAV,RPQ,RPQ] in which we allow the input mapping to contain both sound and exact mapping assertions. We do so by showing an EXPSPACE lower bound for a problem that is closely related to the mapping simplification problem.

Consider a finite alphabet Σ and a finite set $\mathcal{V}$ of variables. A *language constraint* is a statement of the form $e_1 \sqsubseteq e_2$, where e_1 and e_2 are regular expressions over $\Sigma \cup \mathcal{V}$. A *language-constraint problem* P is a finite set of language constraints. A solution to P is an assignment $\sigma : \mathcal{V} \to 2^{\Sigma^*}$, assigning a language over Σ to each variable in $\mathcal{V}$ such that $\mathcal{L}(e_1[\sigma]) = \mathcal{L}(e_2[\sigma])$. It is easy to express the universality problem for context-free grammars as a language-constraint problem, which implies that the latter is undecidable. Here we consider *left-handed* language constraint problems, where we allow constraints of the form $e_1 \sqsubseteq e_2$ and $e_1 = e_2$, but require that variables appear only in the left-hand side of the constraint. The technique for the exact version of the LAV mapping-simplification problem can be used to show an EXPSPACE upper bound for solving left-handed language constraint problems. We now show a matching lower bound.

To prove the result we exploit a reduction from tiling problems [51,52]. A *tile* is a unit square of one of several types and the *tiling problem* we consider is specified by means of a finite set Δ of tile types, two binary relations H and P over Δ, representing horizontal and vertical adjacency relations, respectively, and two distinguished tile types $t_S, t_F \in \Delta$. The tiling problem here consists in determining whether, for a given number n in unary, a region of the integer plane of size $2^n \times k$, for some $k > 0$, can be tiled consistently with the adjacency relations H and P, and with the left bottom tile of the region of type t_S and the right upper tile of type t_F. We also require that the last tile of a row and the first tile of the next row are consistent with H. Using a reduction from acceptance of EXPSPACE Turing machines analogous to the one in [51], it can be shown that this tiling problem is EXPSPACE-complete.

Theorem 6.14. *Solving left-handed language constraints is* EXPSPACE-*complete.*

Proof. Let $T = (\Delta, H, P, t_S, t_F)$ be an instance of the EXPSPACE-complete tiling problem above and n a number in unary. The alphabet is $\Sigma = \Delta \cup \{0,1\}^3 \cup \{\#\}$. Intuitively, the letters in Δ denote tiles, symbols in $\{0,1\}^3$ denote address bits, and # denotes a separation marker. The idea is to encode each tiled cell by a word of length $n+2$ of the form $\# \cdot (\{0,1\}^3)^n \cdot \Delta$, consisting of a marker, an n-bit address, and a tile symbol. We use an element in $\{0,1\}^3$ for each address bit to make it easy to check that two n-bit addresses are consecutive; we use n-bits for the current address, n bits for the carry, and n bits for the next address. Thus, each tiling can be described by a word in $(\# \cdot (\{0,1\}^3)^n \cdot \Delta)^*$, obtained by encoding each cell as described above, and then concatenating the symbols, first column by column and then row by row.

Consider a word $w \in \Sigma^*$. Such a word does *not* describe a proper tiling if one of the following *errors* can be found in the word:

1. The symbol # does not occur precisely in positions $(n+2)i$, for $i = 0, 1, \ldots.$
2. The symbols in Δ do not occur precisely in positions $n+1+(n+2)i$, for $i = 0, 1, \ldots.$
3. The first address is not 0^n.
4. The last address is not 1^n.
5. There is a pair of adjacent but not successive addresses.
6. The first tile is not t_S.
7. The last tile is not t_F.
8. There is a pair of adjacent blocks with tiles that violate the relation H.
9. There is a pair of *vertically adjacent* blocks with tiles that violate the relation P.

We do need to define the notion of vertical adjacency. Two blocks are vertically adjacent if their addresses agree and either both addresses are 0^n and there is no occurrence of 0^n between them, or both addresses are not 0^n and there is precisely one occurrence of 0^n between them.

If the tiling problem has no solution, then every word in Σ^* must contain an error. Conversely, if the tiling problem has a solution, then that solution can be described by a word with no errors. We now define a constraint of the form $e_{error} = \Sigma^*$, where the "task" of e_{error} is to discover errors in candidate words. The expression e_{error} is the sum of several terms corresponding to the various errors. We now sketch how to "discover" these possible errors. In order to have the left-hand sides use only variables, we introduce a variable v_a for each letter $a \in \Sigma$, accompanied by the constraint $v_a = a$. We use $\mathcal{V}_\Sigma$ to abbreviate $\sum_{a \in \Sigma} v_a$.

Most of the errors can be discovered with a single regular expression. For example, the error where the symbol # does not occur precisely in positions $(n+2)i$, for $i = 0, 1, \ldots$, is described using the expression

$$\left(\mathcal{V}_\Sigma^{n+2}\right)^* \cdot \left(\sum_{1 \leqslant i \leqslant n+1} \mathcal{V}_\Sigma^i\right) \cdot \# \cdot \mathcal{V}_\Sigma^*.$$

We encode addresses by words in $(\{0,1\}^3)^n$, which consist of three n-bit words: the first word is an n-bit address, the second word is the successor of that address, and the third word is the sequence of n carry bits. The expression e_{error} contains a term that finds errors in words of the form $(\{0,1\}^3)^n$ used in the encoding candidate solutions. To ensure that adjacent addresses are successive, e_{error} contains, for $0 \leqslant i \leqslant n-1$, the expression

$$\begin{aligned}&\left(\mathcal{V}_\Sigma^{n+2}\right)^* \cdot \# \cdot \left(\{0,1\}^3\right)^i \cdot \{0,1\} \times \{0\} \times \{0,1\} \cdot \left(\{0,1\}^3\right)^{n-i-1} \cdot \Delta\\ &\quad \cdot \# \cdot \left(\{0,1\}^3\right)^i \cdot \{1\} \times \{0,1\}^2 \cdot \left(\{0,1\}^3\right)^{n-i-1} \cdot \left(\mathcal{V}_\Sigma^{n+2}\right)^*\end{aligned}$$

as well as the expression

$$\begin{aligned}&\left(\mathcal{V}_\Sigma^{n+2}\right)^* \cdot \# \cdot \left(\{0,1\}^3\right)^i \cdot \{0,1\} \times \{1\} \times \{0,1\} \cdot \left(\{0,1\}^3\right)^{n-i-1} \cdot \Delta\\ &\quad \cdot \# \cdot \left(\{0,1\}^3\right)^i \cdot \{0\} \times \{0,1\}^2 \cdot \left(\{0,1\}^3\right)^{n-i-1} \cdot \left(\mathcal{V}_\Sigma^{n+2}\right)^*.\end{aligned}$$

These expressions compare corresponding bits in adjacent addresses; the successor of the first address has to agree with the second address.

The one error that is challenging is where there is a pair of *vertically adjacent* blocks with tiles that violate the relation P. Discovering this error is more difficult and cannot be done by one regular expression; rather, several additional constraints are needed. For simplicity we ignore here the fact that each address bit is encoded by three bits rather than one.

Let e_{nza} be a regular expression that describes non-zero addresses: $\sum_{0 \leqslant i \leqslant n-1} \{0,1\}^i \cdot 1 \cdot \{0,1\}^{n-i-1}$.

We add to e_{error} the following term, which discovers non-matching tiles at zero-addressed vertically adjacent tiles:

$$\sum_{(t,t') \notin P} \left(\mathcal{V}_\Sigma^* \cdot \# \cdot 0^n \cdot t \cdot (\# \cdot e_{nza} \cdot \Delta)^* \cdot \# \cdot 0^n \cdot t' \cdot \mathcal{V}_\Sigma^*\right).$$

We need to deal with non-zero-addressed vertically adjacent blocks. For this we use several constraints. First:

$$v_{nzava} \sqsubseteq \sum_{(t,t') \notin P} \left(\# \cdot e_{nza} \cdot t \cdot (\# \cdot e_{nza} \cdot \Delta)^* \cdot \# \cdot 0^n \cdot \Delta \cdot (\# \cdot e_{nza} \cdot \Delta)^* \cdot \# \cdot e_{nza} \cdot t'\right).$$

This says that v_{nzava} describes sequences of blocks that start and end with a pair of non-matching non-zero-addressed blocks, with a single zero-addressed block in between. We still have to impose the constraint that the first and last block have equal addresses. We do this with n constraints, one for each bit of the address. That is for each i, $0 \leqslant i \leqslant n-1$, we add the constraint:

$$\begin{aligned}v_{nzava} \sqsubseteq\ &\left(\# \cdot \{0,1\}^i \cdot 0 \cdot \{0,1\}^{n-i-1} \cdot \Delta \cdot \left(\# \cdot \{0,1\}^n \cdot \Delta\right)^* \cdot \# \cdot \{0,1\}^i \cdot 0 \cdot \{0,1\}^{n-i-1} \cdot \Delta\right)\\ &+ \left(\# \cdot \{0,1\}^i \cdot 1 \cdot \{0,1\}^{n-i-1} \cdot \Delta \cdot \left(\# \cdot \{0,1\}^n \cdot \Delta\right)^* \cdot \# \cdot \{0,1\}^i \cdot 1 \cdot \{0,1\}^{n-i-1} \cdot \Delta\right).\end{aligned}$$

This constraint says that the i-th bits of the first and last addresses are either both 0 or both 1.

Now we can add to e_{error} the term $\mathcal{V}_\Sigma^* \cdot v_{nzava} \cdot \mathcal{V}_\Sigma^*$, which discovers all errors due to not-matching, non-zero-addressed vertically adjacent blocks.

Note that the constraint system constructed is of size quadratic in the size of the tiling system. If the tiling problem has no solution, then every word in Σ^* contains an error and the constraint problem constructed is satisfiable. If the tiling problem has a solution, then a word describing a proper tiling has no error, and for no assignment $\sigma : \mathcal{V} \to 2^{\Sigma^*}$ we have $\mathcal{L}(e_{error}[\sigma]) = \Sigma^*$, since e_{error} captures only errors. □

Let MSimp[mixed,lav,rpq,rpq] be the following decision problem: given an rpq-based schema mapping M consisting both of sound and of exact mapping assertions, check whether there exists an rpq-based lav schema mapping M' of type exact such that $M' \models M$ and $M' \not\models_{triv} M$.

As a corollary of Theorem 6.14, we get the following result.

Corollary 6.15. MSimp[mixed,lav,rpq,rpq] *is* ExpSpace*-complete.*

We conjecture that MSimp[exact,lav,rpq,rpq] is also ExpSpace-complete.

Our results on simplification for RPQs are summarized in Table 1.

7. Extensions

In this section we sketch the extension of the results of the previous section on simplification in terms of lav mappings to more expressive classes of queries: 2RPQs and their conjunctions.

7.1. 2RPQs

Consider now simplification for mappings based on 2RPQs, expressed by means of 1NFAs over the alphabets $\Sigma_s^{\pm}$ and $\Sigma_t^{\pm}$.

A key concept for 2RPQs is that of *folding* of a language [22], which intuitively denotes the set of words that are the result of repeatedly canceling out adjacent occurrences of a symbol and its inverse. Let $u, v \in \Sigma^{\pm}$. We say that v *folds* onto u, denoted $v \rightsquigarrow u$, if v can be "folded" on u, e.g., $abb^{-}bc \rightsquigarrow abc$. Formally, we say that $v = v_1 \cdots v_m$ folds onto $u = u_1 \cdots u_n$ if there is a sequence $i_0, \dots, i_m$ of positive integers between 0 and $|u|$ such that

- $i_0 = 0$ and $i_m = n$, and
- for $j \in \{0, \dots, m\}$, either $i_{j+1} = i_j + 1$ and $v_{j+1} = u_{i_j+1}$, or $i_{j+1} = i_j - 1$ and $v_{j+1} = u^{-}_{i_{j+1}}$.

Let L be a language over $\Sigma^{\pm}$. We define $fold(L) = \{u \mid v \rightsquigarrow u, v \in L\}$.

A language-theoretic characterization for containment of 2RPQs was provided in [22]:

Lemma 7.1. *Let q_1 and q_2 be 2RPQs. Then $q_1 \sqsubseteq q_2$ iff $\mathcal{L}(q_1) \subseteq fold(\mathcal{L}(q_2))$.*

Furthermore, it is shown in [22] that if A is an n-state 1NFA over $\Sigma^{\pm}$, then there is a 2NFA A^f over Σ for $fold(\mathcal{L}(A))$ with $n \cdot (|\Sigma^{\pm}| + 1)$ states. (We use 2NFA to refer to two-way automata.)

In the mapping simplification problem, we are given 2RPQs q_s and q_t, expressed as 1NFAs A_s and A_t, respectively, and we are asked whether there exists a 2RPQ-based LAV mapping M_L such that $q_s[M_L] \sqsubseteq q_t$ or $q_s[M_L] = q_t$, and also $q_s[V] \not\equiv \emptyset$.

The approach using congruence classes described above applies also to 2RPQs. A simplistic approach would be to convert the 2NFA for $fold(\mathcal{L}(A_t))$ into a 1NFA, with an exponential blow-up, and proceed as in Section 6. To avoid this exponential blowup, we need an exponential bound on the number of congruence classes. For a 1NFA, we saw that each congruence class can be defined in terms of a binary relation over its set of states. It turns out that for a 2NFA A, a congruence class can be defined in terms of *four* binary relations over the set S_t of states of A:

1. R_{lr}: a pair $(s_1, s_2) \in R_{lr}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the left and exited on the right.
2. R_{rl}: a pair $(s_1, s_2) \in R_{rl}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the right and exited on the left.
3. R_{ll}: a pair $(s_1, s_2) \in R_{ll}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the left and exited on the left.
4. R_{rr}: a pair $(s_1, s_2) \in R_{rr}$ means that there is a word w that leads A from s_1 to s_2, where w is entered on the right and exited on the right.

That is, every choice of four binary relations over the state space of A^f corresponds to a congruence class with respect to the language of A^f. Thus, the number of congruence classes when A^f has m states is 2^{4m^2} rather than 2^{m^2}, which is still an exponential. This enables us to adapt the technique of Section 6 with essentially the same complexity bounds.

Theorem 7.2. MSIMP[SOUND,LAV,2RPQ,2RPQ] *is* PSPACE-*complete.*
MSIMP[EXACT,LAV,2RPQ,2RPQ] *is in* EXPSPACE.

7.2. Conjunctions of 2RPQs and their unions

Finally, we consider the mapping simplification problem for conjunctions of 2RPQs and their unions, abbreviated (U)C2RPQs [53], which are (unions of) conjunctive queries constructed from binary atoms whose predicate is a 2RPQ. Specifically, a C2RPQ q of arity n is written in the form

$$\{(x_1, \dots, x_n) \mid q_1(y_1, y_2) \wedge \cdots \wedge q_m(y_{2m-1}, y_{2m})\}$$

where $x_1, \dots, x_n, y_1, \dots, y_{2m}$ range over a set $\{z_1, \dots, z_k\}$ of variables, $\{x_1, \dots, x_n\} \subseteq \{y_1, \dots, y_{2m}\}$, and each q_j is a 2RPQ. When evaluated over a database $\mathcal{D}$ over Σ, the C2RPQ q computes the set of tuples $(o_1, \dots, o_n)$ of objects such that there is a total mapping φ from $\{z_1, \dots, z_k\}$ to the objects in $\mathcal{D}$ with $\varphi(x_i) = o_i$, for $i \in \{1, \dots, n\}$, and $(\varphi(y_{2j-1}), \varphi(y_{2j})) \in q_j^{\mathcal{D}}$, for $j \in \{1, \dots, m\}$.

Consider first the mapping simplification problem for the case where the input mapping is expressed in terms of CRPQs, where the constituent RPQs are expressed by means of 1NFAs, over the alphabets $\Sigma_s^{\pm}$ and $\Sigma_t^{\pm}$. Here the LAV mappings have to be in terms of RPQs, rather than CRPQs, since CRPQs are not closed under substitutions. The crux of our approach is to reduce containment of two CRPQs, q_1 and q_2 to containment of standard languages. This was done in [53]. Let q_h, for $h = \{1, 2\}$, be in the form

$$q_h = \{(x_1, \dots, x_n) \mid q_{h,1}(y_{h,1}, y_{h,2}) \wedge \cdots \wedge q_{h,m_h}(y_{h,2m_h-1}, y_{h,2m_h})\}$$

Table 1
Summary of complexity results on mapping simplification.

Type	Form	CQ-CQ	UCQ-CQ	UCQ-UCQ	(2)RPQ-(2)RPQ	UC2RPQ-2RPQ
SOUND	LAV	NP-c	NP-c	NP-c	PSpace-c	ExpSpace
	GAV	NP	NP	NP		
EXACT	LAV/GAV	NP-c	NP-c	Π_2^p	ExpSpace PSpace-hard	2ExpSpace
MIXED	LAV	NP-c	NP-c	Π_2^p	ExpSpace-c	2ExpSpace

and let $\mathcal{V}_1$, $\mathcal{V}_2$ be the sets of variables of q_1 and q_2 respectively. It is shown in [53] that the containment $q_1 \sqsubseteq q_2$ can be reduced to the containment $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ of two word automata A_1 and A_2. A_1 is a 1NFA, whose size is exponential in q_1 and it accepts certain words of the form

$$\$d_1 w_1 d_2 \$ d_3 w_2 d_4 \$ \cdots \$ d_{2m_1-1} w_{m_1} d_{2m_1} \$$$

where each d_i is a subset of $\mathcal{V}_1$ and the words w_i are over the alphabet of A_1. Such words constitute a linear representation of certain graph databases that are canonical for q_1 in some sense. A_2 is a 2NFA, whose size is exponential in the size of q_2, and it accepts words of the above form if the there is an appropriate mapping from q_2 to the database represented by these words. The reduction of the containment $q_1 \sqsubseteq q_2$ to $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$ is shown in [53].

The ability to reduce containment of CRPQs to containment of word automata means that we can also apply the congruence-class technique of Section 6. Suppose that we have an RPQ-based LAV mapping M_L such that $\mathcal{L}(A_s[M_L]) \subseteq \mathcal{L}(A_t)$. Then we can again assume that the queries in the LAV mapping are closed with respect to the congruence classes of A_t. Thus, the techniques of Section 6 can be applied, and one can show that MSimp[SOUND,LAV,CRPQ,RPQ] is in ExpSpace and MSimp[EXACT,LAV,CRPQ,RPQ] is in 2ExpSpace.

Finally, dealing with UC2RPQ queries, which combine UCQs and 2RPQs, requires combining the techniques developed for RPQs, 2RPQs, and CRPQs. The key idea is the reduction of query containment to containment of word automata. The resulting upper bounds are identical to those for CRPQs.

8. Conclusions

We have introduced the problem of simplifying schema mappings based on logical implication. The problem comes in different forms, depending on the type of simplification to achieve, on whether the mappings are sound or exact, and on the types of queries used in the mappings. We have provided a formalization of the problem, and we have presented techniques and complexity bounds for both relational and graph databases. We have concentrated on LAV simplification, and we have discussed the GAV case only for relational schema mappings. Our results are summarized in Table 1. There are a number of results left open by our investigation. In particular, GAV simplification and synthesis for RPQs is largely open. Moreover, in LAV tight lower-bounds are missing for the cases of UCQs and RPQs. We observe that the techniques we have presented in Sections 5, 6, and 7 to prove our upper bounds for LAV simplification in the sound and exact cases are based on performing suitable guesses and containment checks independently for each input mapping assertion. Hence, they extend easily to the mixed case, providing the same upper bounds as for the exact case, which justifies the entries in the last row of the table.

In the future, we plan to continue investigating schema mapping simplification along different directions. In particular, we aim at addressing GAV simplification for graph databases, and we plan to study schema mapping simplification for tree-based (e.g., XML) semi-structured data.

Acknowledgments

The work is partially supported by the EU under the project ACSI (Artifact-Centric Service Interoperation), grant No. FP7-257593, by Regione Lazio under the project “Integrazione semantica di dati e servizi per le aziende in rete”, and by NSF grants CCF-0613889, CCF-0728882, and CNS 1049862.

References

[1] M. Lenzerini, Data integration: A theoretical perspective, in: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), 2002, pp. 233–246.

[2] A.Y. Halevy, A. Rajaraman, J. Ordille, Data integration: The teenage years, in: Proc. of the 32nd Int. Conf. on Very Large Data Bases (VLDB 2006), 2006, pp. 9–16.

[3] P.G. Kolaitis, Schema mappings, data exchange, and metadata management, in: Proc. of the 24th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005), 2005, pp. 61–75.

[4] P. Barcelò, Logical foundations of relational data exchange, SIGMOD Record 38 (1) (2009) 49–58.

[5] P.A. Bernstein, H. Ho, Model management and schema mappings: Theory and practices, in: Proc. of the 33rd Int. Conf. on Very Large Data Bases (VLDB 2007), 2007, pp. 1439–1440.

[6] J.D. Ullman, Information integration using logical views, in: Proc. of the 6th Int. Conf. on Database Theory (ICDT'97), in: Lecture Notes in Comput. Sci., vol. 1186, Springer, 1997, pp. 19–40.

[7] B. ten Cate, P.G. Kolaitis, Structural characterizations of schema-mapping languages, in: Proc. of the 12th Int. Conf. on Database Theory (ICDT 2009), 2009, pp. 63–72.

[8] B. Alexe, P. Kolaitis, W.-C. Tan, Characterizing schema mappings via data examples, in: Proc. of the 29th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2010), 2010, pp. 261–271.

[9] R. Fagin, P.G. Kolaitis, A. Nash, L. Popa, Towards a theory of schema-mapping optimization, in: Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2008), 2008, pp. 33–42.

[10] G. Gottlob, R. Pichler, V. Savenkov, Normalization and optimization of schema mappings, Proc. VLDB Endowment 2 (1) (2009) 1102–1113.

[11] R. Fagin, P.G. Kolaitis, L. Popa, W.C. Tan, Reverse data exchange: Coping with nulls, in: Proc. of the 28th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2009), 2009, pp. 23–32.

[12] M. Arenas, R. Fagin, A. Nash, Composition with target constraints, in: Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010), 2010, pp. 129–142.

[13] A. Fuxman, P.G. Kolaitis, R.J. Miller, W.C. Tan, Peer data exchange, ACM Trans. Database Systems 31 (4) (2005) 1454–1498.

[14] G. Grahne, A.O. Mendelzon, Tableau techniques for querying information sources through global schemas, in: Proc. of the 7th Int. Conf. on Database Theory (ICDT'99), in: Lecture Notes in Comput. Sci., vol. 1540, Springer, 1999, pp. 332–347.

[15] M. Arenas, L. Libkin, XML data exchange: Consistency and query answering, in: Proc. of the 24th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005), 2005, pp. 13–24.

[16] I.F. Cruz, A.O. Mendelzon, P.T. Wood, A graphical query language supporting recursion, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1987, pp. 323–330.

[17] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: from Relations to Semistructured Data and XML, Morgan Kaufmann, 2000.

[18] E. Franconi, S. Tessaris, The logic of RDF and SPARQL: a tutorial, in: S. Vansummeren (Ed.), Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006), 2006, p. 355.

[19] P. Barcelò, C.A. Hurtado, L. Libkin, P.T. Wood, Expressive languages for path queries over graph-structured data, in: Proc. of the 29th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2010), 2010, pp. 3–14.

[20] B. Selman, H. Kautz, Knowledge compilation and theory approximation, J. ACM 43 (2) (1996) 193–224.

[21] M. Schaerf, M. Cadoli, Tractable reasoning via approximation, Artificial Intelligence 74 (2) (1995) 249–310.

[22] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi, Reasoning on regular path queries, SIGMOD Record 32 (4) (2003) 83–92.

[23] R. Fagin, P.G. Kolaitis, R.J. Miller, L. Popa, Data exchange: Semantics and query answering, Theoret. Comput. Sci. 336 (1) (2005) 89–124.

[24] R. Fagin, P.G. Kolaitis, L. Popa, Data exchange: Getting to the core, ACM Trans. Database Systems 30 (1) (2005) 174–210.

[25] M. Arenas, P. Barcelo, R. Fagin, L. Libkin, Locally consistent transformations and query answering in data exchange, in: Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004), 2004, pp. 229–240.

[26] A. Fuxman, P.G. Kolaitis, R. Miller, W.C. Tan, Peer data exchange, in: Proc. of the 24th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005), 2005, pp. 160–171.

[27] L. Libkin, C. Sirangelo, Data exchange and schema mappings in open and closed worlds, in: Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2008), 2008, pp. 139–148.

[28] S. Abiteboul, O. Duschka, Complexity of answering queries using materialized views, in: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), 1998, pp. 254–265.

[29] A. Calì, D. Calvanese, G. De Giacomo, M. Lenzerini, Data integration under integrity constraints, Information Systems 29 (2004) 147–163.

[30] A.Y. Halevy, Answering queries using views: A survey, Very Large Database J. 10 (4) (2001) 270–294.

[31] J. Madhavan, A.Y. Halevy, Composing mappings among data sources, in: Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB 2003), 2003, pp. 572–583.

[32] R. Fagin, P.G. Kolaitis, L. Popa, W.-C. Tan, Composing schema mappings: Second-order dependencies to the rescue, ACM Trans. Database Systems 30 (4) (2005) 994–1055.

[33] R. Fagin, Inverting schema mappings, ACM Trans. Database Systems 32 (4) (2007) 25:2–25:53.

[34] R. Fagin, P.G. Kolaitis, L. Popa, W.-C. Tan, Quasi-inverses of schema mappings, ACM Trans. Database Systems 33 (2) (2008) 1–52.

[35] M. Arenas, J. Pérez, C. Riveros, The recovery of a schema mapping: Bringing exchanged data back, ACM Trans. Database Systems 34 (4) (2009) 22:1–22:48.

[36] P.C. Arocena, A. Fuxman, R.J. Miller, Composing local-as-view mappings: Closure and applications, in: Proc. of the 13th Int. Conf. on Database Theory (ICDT 2010), 2010, pp. 209–218.

[37] M. Arenas, J. Pérez, J.L. Reutter, C. Riveros, Foundations of schema mapping management, in: Proc. of the 29th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2010), 2010, pp. 227–238.

[38] F. Baader, R. Küsters, Unification in a description logic with transitive closure of roles, in: R. Nieuwenhuis, A. Voronkov (Eds.), Proc. of the 8th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2001), in: Lecture Notes in Comput. Sci., vol. 2250, Springer, 2001, pp. 217–232.

[39] S. Abiteboul, G. Gottlob, M. Manna, Distributed XML design, in: Proc. of the 28th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2009), 2009, pp. 247–258.

[40] W. Martens, M. Niewerth, T. Schwentick, Schema design for XML repositories: Complexity and tractability, in: Proc. of the 29th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2010), 2010, pp. 239–250.

[41] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison–Wesley Publ. Co., 1995.

[42] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi, Rewriting of regular expressions and regular path queries, J. Comput. System Sci. 64 (3) (2002) 443–465.

[43] A.K. Chandra, P.M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: Proc. of the 9th ACM Symp. on Theory of Computing (STOC'77), 1977, pp. 77–90.

[44] Y. Sagiv, M. Yannakakis, Equivalences among relational expressions with the union and difference operators, J. ACM 27 (4) (1980) 633–655.

[45] P. Buneman, Semistructured data, in: Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97), 1997, pp. 117–121.

[46] S. Abiteboul, Querying semi-structured data, in: Proc. of the 6th Int. Conf. on Database Theory (ICDT'97), 1997, pp. 1–18.

[47] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi, Query processing using views for regular path queries with inverse, in: Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000), 2000, pp. 58–66.

[48] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, D. Srivastava, Answering queries using views, in: Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95), 1995, pp. 95–104.

[49] J.-E. Pin, Syntactic semigroups, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Language Theory, vol. 1, Springer, 1997, pp. 679–746. Ch. 10.

[50] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi, View synthesis from schema mappings, CoRR Technical Report abs/1003.1179, arXiv.org e-Print archive, available at http://arxiv.org/abs/1003.1179, Mar. 2010.

[51] P. van Emde Boas, The convenience of tilings, in: A. Sorbi (Ed.), Complexity, Logic, and Recursion Theory, in: Lect. Notes Pure Appl. Math., vol. 187, Marcel Dekker Inc., 1997, pp. 331–363.

[52] R. Berger, The undecidability of the dominoe problem, Mem. Amer. Math. Soc. 66 (1966) 1–72.

[53] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi, Containment of conjunctive regular path queries with inverse, in: Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000), 2000, pp. 176–185.